# Altova MobileTogether Designer

ALTOVA®
MobileTogether®
DESIGNER

## User & Reference Manual

# Altova MobileTogether Designer
# User & Reference Manual

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Published: 2025

# Table of Contents

# 1 Welcome to MobileTogether Designer

MobileTogether Designer is an entirely free-to-use product for Windows machines that establishes your mobile solutions precisely the way you want them. With an easy-to-comprehend approach, you employ drag-and-drop functionality to create elegant mobile solutions. MobileTogether Designer comes equipped with a complete mobile simulator so that you can instantly simulate your mobile solution in the designer. You can also run the mobile solution directly on your mobile device to view your project in real-time.

***Current version: 10.1***

## MobileTogether Designer tutorials, video demos, and example files

The following resources will help you to get started with MobileTogether Designer:

- Tutorials[72] that take you from the basics to more advanced features
- Video demos that provide a fast introduction to the powerful features of MobileTogether Designer and show you how to build different types of MobileTogether solutions
- Several example files. You can open these files, simulate their workflow, and explore their design features. These files are packaged with MobileTogether Designer and are located in the *Altova/MobileTogetherDesigner* subfolder of the *(My) Documents*[60] folder.

## This documentation

This documentation is the user manual of MobileTogether Designer. It is organized into the following sections:

- New Features 26
- Introduction 60
- Tutorials 72
- User Interface 251
- Project 286
- Page Sources (Data Sources) 315
- Pages and Page Events 380
- Controls and Control Events 403
- Actions 667
- Databases 952
- Design Components 1061
- XPath/XQuery Expressions and Functions 1243
- Global Variables 1298
- Presentation 1311
- Altova Global Resources 1334
- Subprojects and Modules 1348
- Simulation 1355
- MT Debugger 1388
- Automated Testing 1399
- Offline Usage 1416
- Embedded Webpage Solutions 1427
- AppStore Apps 1471
- In-App Purchases 1502
- MT Solutions in UWP Apps 1534
- Server Services 1543
- Server Action Libraries 1551
- Menu Commands 1561
- Frequently Asked Questions 1686
- Appendices 1687

## Latest Documentation

The latest documentation is available online at the Altova website. The online documentation is constantly updated and could contain updates that are not included in the Help that is packaged with the software. Please compare the *Last updated* dates (*see below*) of the packaged and online versions to check whether the online version is a later version.

*Last updated: 13 May 2025*

**Altova website:** App development, Enterprise apps, Enterprise app development, RMAD, Low code app development

# 2 New Features

Given below are the new-features lists of **Version 10** releases.

## Version 10.1

- When generating code for AppStore Apps 1471, privileges have been added for reading media 1472 on Android devices.
- When sending content via the Execute REST Request 837 action, you can select a file or generate the content with an XPath expression or supply Base64 content.
- Images can be embedded in the design file 1090 at the time an image is selected.
- Support fro Android 15.
- For the In-App Purchase 1502 feature, support has been added for Billing Library 7.1.1 on Android devices
- Support for the following additional databases: MySQL 9, SQLite 3.47.2, IBM DB2 12.x. See the topic *Databases* 952 for the full list of supported databases.

## Version 10.0

*External barcode scanners*

Barcodes can be scanned by external barcode scanners (Zebra, Zebra Mobile Computer, and Datalogic) and passed to a MobileTogether solution. See the topic Barcode Scanners 1145 for an overview of how barcode scanners work with MobileTogether and how to design your solutions for barcode scanners.

The following design components have been added to enable the Barcode Scanner feature:

- Actions to connect/disconnect and configure the different scanners: Zebra Connect/Disconnect 772, Zebra Configure 773, Zebra Mobile Computer Connect/Disconnect 775, Zebra Mobile Computer Configure 777, Datalogic Connect/Disconnect 778, and Datalogic Configure 779.
- Global variables 1300 that provide information about MT-client device functionality related to barcode scanning: **`$MT_BluetoothAvailable`**, **`$MT_BluetoothLEAvailable`**, **`$MT_ZebraMobileComputerAvailable`**, and **`$MT_DatalogicScannerAvailable`**.
- MobileTogether extension XPath functions 1262 to provide functionality for the barcode-scanning mechanism: **`mt-bluetooth-started()`**, **`mt-zebra-scanner-connected()`**, **`mt-zebra-scanner-id()`**.
- Page source trees 1145 to store the scanner data and barcode data received from the respective scanners: **`$MT_ZEBRASCANNER`**, **`$MT_ZEBRAMOBILECOMPUTER`**, **`$MT_DATALOGICSCANNER`**.
- Actions for barcode scanner events. The **`OnDataReceived`** event enables you to define what actions to execute when barcode-scanned data is received in the solution. For Zebra scanners, actions can be defined for two additional events: **`OnConnectionEstablished`** and **`OnConnectionTerminated`**. The Actions dialog for these events is accessed via the project property Barcode Scanner Actions 296.
- Simulations now include options for simulating barcode scans. Default settings for some of these options can be set in the Simulation 1 tab 1668 of the Options tab..

*Controls*

- Designers can save embedded images 1090 to a file via the context menu of the Image control. For details, see the Embed Image property of the Image control 541.
- To have the dropdown list of a combo box 459 open automatically when the user tabs to it, set the Browser CSS Class property of the combo box to a value of mt-combo-open-on-focus. To apply the

behavior to multiple combo boxes, this property value can be set on a Table[614] or Page[384] that contains combo boxes.
- A control's Control Variables[404] are defined for individual controls and are evaluated when the control is called. Control variables provide more flexibility to set values according to the context of a specific control.
- The context menu of a Placeholder Control[568] now contains the Replace Placeholder with Template Content[404] command. On selecting the command, the design components of the placeholder's related control template[1200] will replace the placeholder.
- The new HTML Label[529] control enables the formatting of text in the control to vary. The text of the control is a string marked up with HTML markup. On client devices, the text is formatted according to the HTML markup.

*Databases*

- Support for the following additional databases: MySQL 8.2 and 8.3, PostgreSQL 16, MariaDB 11.2, SQLite 3.45. See the topic *Databases*[952] for the full list of supported databases.

*Miscellaneous*

- The Styles Inspector[1332] can be opened during simulations to provide an overview of the computed styles of the controls of the current page.
- To cover the case of solutions on servers that are behind proxy or reverse proxy servers, the extension function[1262] **`mt-client-ip-address()`** has been enhanced to return the value of relevant HTTP headers if such a header is present in the HTTP request. Otherwise the client's IP address is returned.
- The Files Pane[259] has been enhanced: multiple deployable files can be selected for addition to a project and the option to deploy or not has been simplified.
- In Design View, the context menu of design components has been extended to provide options for deploying files[259] and embedding images[1090].
- The command List Unused Functions, User Variables, StyleSheets, Action Groups[1613] has been extended to also list unused localized strings.

## 2.1 Version 9

Given below are the new features of **Version 9.0**.

*MQTT*

- A MobileTogether solution can take part in an MQTT network as a publisher, subscriber, or both. The mechanism by which this participation is enabled is described in the section MQTT 1135.
- Two new MQTT-related actions 761 have been introduced: Publish MQTT Message 763 and (Un) Subscribe to MQTT Topic 764.
- A **$MT_MQTT** 1138 page source has been added. It provides a data source in which message data can be stored..
- The **OnMQTTReceive** 399 page event is triggered when an MQTT message is received for a subscription on that page. An action sequence can also be defined at the solution level 296 for messages received on any page.
- You can also create server services for MQTT actions 1139. This enables you to silently run MQTT actions from MobileTogether Server. To accommodate MQTT, server services functionality 1543 has been extended to include actions executed on receiving an MQTT message.
- For testing actions that are performed when an MQTT message is received, you can specify, in the Options dialog 1669, a file that contains message data.

*Broadcast*

- A MobileTogether solution can take part in a MobileTogether broadcast network as a publisher, subscriber, or both. The mechanism by which this participation is enabled is described in the section Broadcasts 1142.
- The mechanism underlying the Broadcast feature uses two actions: Publish Broadcast Message 768 and (Un)Subscribe Broadcast Topic 768.
- When a solution receives a broadcast message, the message is stored in a dynamic variable named **$MT_Broadcast** 1304.
- The actions to perform when a solution receives a broadcast message are specified in the page event OnBroadcastReceive 400 and/or the Broadcast Actions 296 of the project.
- For testing actions that are performed when a broadcast message is received, you can specify, in the Options dialog 1669, a file that contains message data.

*Actions*

- The Switch and Case actions 895 enable you to choose one set from several sets of actions to execute.
- Two new MQTT-related actions 761 have been introduced: Publish MQTT Message 763 and (Un) Subscribe to MQTT Topic 764.
- Two new Broadcast-related actions 1142 have been introduced: Publish Broadcast Message 768 and (Un) Subscribe Broadcast Topic 768.

*Databases*

- When saving databases 1035, a new option is available that enables all rows to be saved if anything in the table has been changed. The option is available in Save actions 803 and the DB's page source context menu 359.
- When adding related tables 1024 to a DB structure, an option has been added to list tables even if they contain no rows.

- Native support has been added for MySQL and MariaDB. See Databases 952 for details about DB support.

*Controls*

- Table controls 614 can be designed with more structural flexibility and have a more varied mix of static and dynamic rows. Dynamic tables with repeating rows 1069 can be converted to repeating tables 1065 and have more than one table row group.
- In top-level Tables 614 that contain row groups (for example, a row group of `Person` rows), actions can be executed when the end user makes a gesture 1078 on an individual row. A gesture on a table row 1078 may be one of the following: (i) swiping left or right 1078 or (ii) dragging 1078 (typically to a new position within the row group).
- Two dynamic local variables related to the dragging-and-dropping of table rows have been added: **`$MT_DragAndDropSource`** 1304 and **`$MT_DragAndDroptarget`** 1300.
- Button control 417: Additional images have been added to the library of images that can be displayed on buttons (see the `Button Image` property).

*InApp Purchases and AppStore Apps*

- The InApp Purchases 1502 functionality has been updated to accommodate newer Android billing specifications.
- Input parameters for AppStore Apps 1471 can be specified via the app's SPL template 1492, as a property of the SPL template's $Options object 1497.

*Variables and Functions*

- User variables 1309 can now also be stored for server-only use. This option for user variables is in addition to those for client-only variables and variables that are available on both client and server.
- The MobileTogether Extension Function 1262 **`mt-has-serveraccess`** has been extended with a second signature that checks whether a client can access a specified server URL.
- The MobileTogether Extension Functions 1262 **`mt-hexBinary-to-string`** and **`mt-string-to-hexBinary`** each has been given a second signature that assumes UTF-8 encoding if no **`Encoding`** argument is supplied.
- When a solution receives a broadcast message, the message is stored in a dynamic variable named **`$MT_Broadcast`** 1304.
- Two dynamic local variables related to the dragging-and-dropping of table rows have been added: **`$MT_DragAndDropSource`** 1304 and **`$MT_DragAndDroptarget`** 1300.
- A new dynamic variable named **`$MT_UserMail`** 1304 has been added to hold the email address of the user currently used for server communications.

*Interface*

- You can select a classic, light, or dark theme 1681 for the application.
- The Help system 1683 has been reorganized to provide Online Help by default, with an option to use the locally installed PDF user manual 1679 as the alternative default.

*Miscellaneous*

- During simulations, the page source structure can be modified in the simulator's page sources pane 1356 (by adding new elements and attributes and/or by renaming elements and attributes). This enables you to try out different page source structures during simulation.

- When defining REST requests for an HTTP connection[328], you can specify connection and request timeouts.
- A new dynamic variable named **$MT_UserMail** has been added to hold the email address of the user currently used for server communications.
- A control template[1200] used to be able to be overridden by actions set on its parent placeholder control[1208], but not by actions set on any higher-level ancestor placeholder controls if such overrides existed. From this release onwards, the overrides start at the outermost placeholder control and continue to be executed for each placeholder control lower in the hierarchy that has overrides defined for them.
- Trial Runs on Clients[1370] have been enhanced to enable text searches[1403], including in trial runs on clients for Automated Testing[1403].
- Settings have been added to define network settings[1675], the XPath Debugger display[1678], and the default Help format[1679].
- Support for Android 14.
- Support for the following additional databases: PostGreSQL 15.1, Microsoft SQL Server 2022, Firebird 4. See the topic *Databases*[952] for the full list of supported databases.

## 2.2 Version 8

Given below are the new-features lists of **Version 8** releases.

### Version 8.1

*Progress Indicator feature*

A number of new components (*listed below*) have been added to implement this feature. For an overview of how they work together, see the Progress Indicator tutorial [241].

- A Progress Show Subpage action [795], which (i) specifies the subpage that will be displayed on the client to indicate the progress of server actions; and (ii) defines, as its child actions, the server actions to carry out for which the progress indicator is required.
- A Progress Update [796] action, which specifies what value to pass to the dynamically responsive **`$MT_Progress`** [1304] variable.
- A **`$MT_Progress`** [1304] global variable holds dynamically changing data about the progress of a specific set of actions on the server. The values that go into the variable are defined in the Progress Update [796] action.
- A page event named **`OnProgressUpdate`** [401], which is triggered by the Progress Update [796] action and can be used to update a progress subpage with information about server-action progress (via the `$MT_Progress` variable).
- A Progress Send Cancellation [797] action, which, when triggered, sets the **`mt-progress-cancellation()`** [1262] function to `true()`.
- A **`mt-progress-cancellation()`** [1262] function, which can be used to test whether the client has sent a cancellation request or not.

*Solution deployment and execution*

- An internal MobileTogether Designer deployment check ensures that a solution containing features supported on MobileTogether Server Advanced Edition will not be deployed to a MobileTogether Server Standard Edition. See *Deploying the Project* [291].
- The Solution Execution [915] action has a new option to restart the solution only when a newer solution exists on the server.
- Ability to start a MobileTogether Server service via URL and specify input parameters for the service. See the topic Start Service via URL [1549].

*Actions*

- The Actions [667] dialog has been reorganized into smaller sections to make the actions easier to find. Furthermore, each section can be expanded/collapsed so that only actions that you use frequently are visible; this also helps to make frequently used actions easier to find.
- The Save [803] action for DBs enables you to choose whether to save data in modified tables or not, and, if data is to be saved, then whether all records in the related table should be replaced or only the data that has been modified.
- The DB Begin Transaction [856] action gets an option to implement the `EXCLUSIVE` transaction of SQLite DBs.
- The Update Variable [903] action has been extended so that variables can accept results from Action Groups and subpages, in addition to those evaluated by XPath expressions.
- The Copy/Paste Clipboard [923] action enables you to copy text to the clipboard and subsequently paste the copied text to a page source node.

*Databases*

- When saving to a DB [803], you can choose whether to save data in modified tables or not, and, if data is to be saved, then whether all records in the related table should be replaced or only the data that has been modified.
- The DB Begin Transaction [856] action gets an option to implement the `EXCLUSIVE` transaction of SQLite DBs.
- If you have manually added nodes to a DB page source [315] and you click the **Reload Structure** [359] context menu command of the page source, then a dialog appears in which you can choose whether to remove or keep the nodes you added.
- Support for the following additional databases: IBM Db2 for i 7.5, PostgreSQL 14.5, MariaDB 10.9.2, SQLite 3.39.2. See the topic *Databases* [952] for the full list of supported databases.

*Styles*

- A *Strikethrough* text-decoration property has been introduced on controls where it is suitable: Label [554], Check Box [447], Radio Button [571], and Button [417].
- If you want the default style values of your project's design components to be as similar as possible, then set the *UI Compatibility Mode* of More Project Settings [296] to **`true`**.

*Localization*

- In the Localization dialog [1600], you can use the **F2** key to start editing—alternatively to double-clicking in a field.
- In the Localization dialog [1600], if the name of a custom string in the default language is modified, then a dialog appears asking whether you want to change the string's name in all XPath expressions where the original string is used.

*Miscellaneous*

- Ability to start a MobileTogether Server service via URL and specify input parameters for the service. See the topic Start Service via URL [1549].
- A validation error [1589] about a missing page source of a page provides a quick-fix for adding the missing page source to not only that page but also other pages where that page source is required but is not present. This enables you to fix missing-page-source-errors faster.
- If you have manually added nodes to an external page source [315] and you click the **Reload Structure** [359] context menu command of the page source, then a dialog appears in which you can choose whether to remove or keep the nodes you added.
- In the Files Pane [259], you can copy a file's absolute path. This is in addition to copying the file path as it appears in the window (relative or absolute).
- When unassigning a page source node from a control [403], the name of the page source node is displayed in the command (in the control's context menu).
- Support for Android 13.
- Support for the following additional databases: IBM Db2 for i 7.5, PostgreSQL 14.5, MariaDB 10.9.2, SQLite 3.39.2. See the topic *Databases* [952] for the full list of supported databases.

## Version 8.0

*Altova RecordsManager*

- Altova RecordsManager [70] is a MobileTogether-based solution that enables users to design and use databases easily and quickly. RecordsManager is installed as a package with your MobileTogether Designer installation. Read more about Altova RecordsManager at its web page.

- In MobileTogether Designer, you can try out RecordsManager[70] by running a simulation of it.
- You can deploy RecordsManager[70] to a MobileTogether Server, and you and your associates can access it from there.
- You can also create an AppStore App of RecordsManager[70], which can then be downloaded and used like any other AppStore App[1471].

### *Themes*

- The Set Theme[930] action can be used to restart the solution with a new theme.
- The Altova extension function[1262] **`mt-client-theme()`** can be used to find out the currently applied theme and the theme set for the solution.

### *Subprojects and Modules*

- The new Subprojects feature[1348] enables projects to be included as subprojects in other projects. This enables a wide range of components that are defined in a project to be re-used across multiple projects. A subproject can itself include another subproject.
- Included subprojects are displayed in the Files Pane[259].
- Commands related to subprojects are available in the Refactor[1610] menu.
- Modules[1352] enable you to group design components in order to apply a common property to these components. Properties that can be applied: (i) *Background color* (which helps to visually locate components of a module in application windows and dialog boxes); (ii) the *Export* property which can be used to determine which components are extracted to subprojects generated from the project[1350].
- A new Modules Pane[263] provides a single location where modules are managed.

### *Server action libraries and Action Groups*

- A server action library[1551] is a new type of solution that defines one or more Action Groups. A solution can now call a server action library's Action Group to execute common tasks (such as sending emails from the server) or obtain a return value computed in the server action library (and not in the calling solution).
- You can manage the server action libraries of a solution in the solution's Files Pane[259].
- Commands related to server action libraries are available in the Refactor[1610] menu.

### *Databases*

- If a DB data source references relational tables, then the referenced tables can be made available automatically[955] as nodes of the page source. These nodes can then be used in the design.
- The options to save data to a DB[1035] have been enhanced to intelligently write data to relationally linked databases.
- The set of mt-db functions[1262] have been enhanced to support relationally linked databases.
- A new **`mt-db-row-from-original`**[1262] function has been introduced.
- A new Switch DB[871] action enables you to switch the database associated with a page source at any point in the workflow.
- Support for the following additional database versions[952]: DB2 11.5.7; MariaDB 10.6.5; PostGreSQL 14; MySQL 8.0.28; SQLite 3.37.2.

### *Deployment to server of solution and packages, and server side solution files*

- Server side solution files can be added to the project in the Files Pane[259]. These files will be deployed and can be updated during the deployment step[1574]. As a result, they do not have to be copied manually to the server.

- If multiple languages are available in the solution, then, during the deployment step[1574], you can specify whether the solution should be opened on a client using the default language of the client, or the default language of the solution, or any of the other languages of the solution.
- During deployment, you can specify that certain actions be carried out on the server. For example, you might want to rename a server file or send an email notification. These actions are defined for the **`OnServerDeployment`** event, which is accessed via the More Project Settings dialog[296]. The **`OnServerDeployment`** actions can also use parameters, the values of which are passed to the solution as input parameters during the deployment procedure[1574]. The input parameters are stored in the **`$MT_InputParameters`**[1300] variable, from where they can be accessed for use in the **`OnServerDeployment`** actions.
- Not only can MobileTogether packages[295] be deployed to the server as a solution, but you can also specify, during the creation of the MobileTogether package[1572], that server side solution files also be deployed together with the package.

*Actions*

- The Set Theme[930] action can be used to restart the solution with a new theme.
- The Scroll To[788] action has been enhanced with a new property that lets you define whether to execute the action immediately on being processed or after all actions of the current event have been processed.
- In the Actions dialog, Action Groups[940] are now managed in a separate pane on the right-hand side of the dialog.
- A new Rename File/Folder[850] action enables you to change the names of files and folders.
- A new Lock/Unlock Clients[919] action enables clients running a solution to be locked form the server while actions are being carried out on the sever. A simulator option, *Prevent Client Lock*, enables the lock to be overridden during simulations[1355].
- A new project property[296] named *Phone Settings Changed* enables you to define a set of actions to execute when a phone setting is changed.
- The Delete File/Folder[853] action provides the ability to move files to the recycle bin.
- The Copy File/Folder[851] action enables you to copy a fie or the contents of a folder to another location.

*Interface*

- In the Page Sources Pane[270], you can add comments via the context menu[359] to the root node and tree nodes of a page source.
- The Files pane[259] now additionally displays subprojects and server side solution files.
- In the Files pane[259], you can open a listed filed in the default application for its file type.
- In the Messages Pane[278], the toolbar contains a new button for pasting error messages reported in the MobileTogether Server log. The pasted message will contain links that help to locate the source of the error in the design.

*Images*

- Base64 images in an SQL database can now also be used as an image source for the Image control[541].
- For loading an image via the Load Image action[707], you can specify whether EXIF images should be auto-rotated according to the corresponding information in the EXIF data.

*Simulations*

- Additional options can be set during simulations[1355]: whether a light or dark theme is used; to lock the client from server access; to display the sequence of tab-ordered controls; to restrict logging to errors only.
- The Run[1617] menu contains a new command that enables the selection of various simulator options[1620].
- In addition to being able to deploy a MobileTogether package to the server from MobileTogether Designer, you can also run a simulation of the package's solution in MobileTogether Designer. See MobileTogether Packages[295].

*Miscellaneous*

- The **`mt-get-page-source-structure()`**[1262] XPath extension function has been extended with a third argument to specify a restricted substructure from that returned by the first argument.
- In a control template, placeholders and actions of controls have a `Prevent Action Override` property. If this property is set to `true`, then any action overrides[1208] that are defined for ancestor placeholders of the control template will be disabled.
- Support for Android 12.
- Support for iOS 15.
- Support for Windows 11.

## 2.3 Version 7

Given below are the new-features lists of **Version 7** releases.

### Version 7.3

*In-app Purchases*

- You can add in-app purchases[1502] to your AppStore Apps[1471]. The mechanism to implement in-app purchases and an included example project are described in the section In-App Purchases[1502]. The components of the mechanism are listed below.
- The new MobileTogether extension functions[1262] `mt-in-app-purchase-product-to-platform` and `mt-in-app-purchase-platform-to-product` retrieve, respectively, a product's ID on a given platform from the submitted product name, and vice versa.
- The new MobileTogether extension function[1262] `mt-in-app-purchase-service-started` can be used to check whether the client device is running its in-app service[1521].
- The main interface between the MT design and app stores is the new In-App-Purchase Page Source[1507], which can dynamically hold app store data about products and purchases.
- The following new actions implement in-app purchases: Purchase[933], Restore Purchases[934], Query Purchases[934], Query Available Products[935], Acknowledge Purchase[936], Get/Report Consumable[937].
- A dynamic local variable[1304] named `$MT_UpdatedInAppPurchases` holds a sequence that comprises the SKU-IDs of the most recently updated purchases.
- The new `OnPurchaseUpdated`[296] event of the In-App Purchase Actions project property[296] enables actions to be specified when the In-App-Purchase Page Source in the design is updated with data about the latest purchase.
- The following MobileTogether extension functions[1262] can be used to check the success of the last request to the app store: `mt-last-in-app-purchase-response-code`, `mt-last-in-app-purchase-response-text`, and `mt-last-in-app-purchase-response-was-user-canceled`.
- To enable the simulation of in-app purchases in MobileTogether Designer, you can use data from an XML file as a substitute for app store data[1669].

*AppStore apps and their simulations*

- Ability to run client-side simulations[1370] of AppStore Apps (compiled apps)[1471]. This is enabled via a new *Build Modes* setting in the first screen for generating program code[1472].
- Simulate trial runs of app-store apps[1370]. During the code-generation procedure[1472], you can specify the connection details of your MobileTogether Designer machine. Once this is done, you can run client simulations of the compiled app at any subsequent time. Even after changing your design, you will not need to re-compile the app to run a client simulation. The app will connect to MobileTogether Designer and use the currently open version of the design.
- When generating program code for compiled apps[1471], you can select whether to use the original SPL template directory or a custom SPL template directory. The option for this selection is available in the fifth screen for generating program code[1472].

*MobileTogether solutions in UWP apps*

- A SolutionView control[1534] is now available. One or more of these controls can be placed in a UWP app and in this way enables one or more MobileTogether solutions to be included in a UWP app.

*Actions*

- New actions to implement the In-app Purchases mechanism: Purchase 933, Restore Purchases 934, Query Purchases 934, Query Available Products 935, Acknowledge Purchase 936, Get/Report Consumable 937.
- The View Image 713 action has been extended with an auto-rotate property that enables the viewed image to be auto-rotated.
- The Open URL 681 action can be used to construct a command line instruction that can be used in designer simulations.

*Controls*

- A new `Tooltip` property has been introduced for several controls 403. A tooltip provides useful information to the end user about the control.
- The Image control 541 has a new property, `Max Control Height`, to set the maximum height of the control as an absolute value. It also has new auto-rotate property.
- The control that currently has the focus will be directly selected when the Localization dialog 1600 is opened.
- In tables, cells can be set to belong to a group and the text in these cells can be auto-sized as a group. This feature has been enhanced with improvements to the table control's `Wrap Content Auto-Fit Group` 614 property.

*Miscellaneous*

- REST requests now support additional verbs 332 from the HTTP vocabulary. This allows requests to be made with verbs other than the commonly used verbs `GET`, `PUT`, `POST`, `DELETE`.
- Previously read-only files could be added to the Files Pane 259 in order to deploy them to the server. Now you can choose whether to deploy such files to the server or client, or both. Deploying a frequently used file directly to the client can save processing time by avoiding have to transfer the file each time from the server.
- The functionality related to deployed files, as these are accessed in the Files Pane 259, has been streamlined into a more compact design.
- Support for Android 11.
- Support for the following additional database versions 952: IBM iSeries 7.4, IBM DB2 11.5, and PostGreSQL 13, MySQL 8.0.25.

## Version 7.2

*Controls*

- Button control 417: Additional images have been added to the library of images that can be displayed on buttons (via the `Button Image` property).
- Button control 417: The new properties `Button Image Color` and `Button Image Color (Disabled)` provide the ability to select a separate color for each of the button's two states (enabled and disabled).
- Horizontal Slider control 520: The new properties `Slider Color`, `Slider Thumb Color`, and `Slider Color (Disabled)` set a separate color for a slider's scale line and marker, and yet another color for the slider (both scale line and marker) when the slider is disabled.
- If controls in the cells of a table 614 have been set to belong to a group having auto-sized text, then these cells can re-sized so that all cells are wrapped into the available view. This setting is made via the table property 614 `Wrap Content Auto-Fit Group`.
- Table cells 614 now have cell padding properties.

*Page sources*

- In the Page Sources window[270], a new context menu command[359] of root nodes deletes the selected page source from all pages of the project.
- In the **Refactor** menu, the command **List Page Sources by Attribute**[1610] lists the project's page sources into groups according to the values of their attributes.
- The default XML file of XML page sources can be edited directly in Altova XMLSpy by selecting the **View Default File in XMLSpy**[359] command.

*Actions*

- The Log Message action[925] enables a customized message to be logged on the server or client during the execution of an action. This helps to analyze the app's behavior during an action.
- This Backup/Restore SQLite DB action[869] enables you to backup an SQLite database to a folder that you designate. Multiple backups are allowed. You can subsequently restore the SQLite database from one of these backups.

*Miscellaneous*

- New MobileTogether extension functions[1262] have been added: (i) `mt-control-text-offset`, (ii) `mt-db-file-path`, (iii) `mt-page-stack`, (iv) `mt-server-variable`.
- The Options dialog provides a setting for enabling the simulation of server variables[1669].
- The *Search in Translations* option of the Localization dialog[1600] enables you to search for text in localized strings.
- In the More Project Settings[296] dialog, which is accessed via the project's properties[296], you can enter the message that you want to have displayed on the client device when the server times out.
- Errors that occur during the execution of actions[667] where error-handling is provided are treated as warnings. The number of reported errors is thus reduced. The advantage is that you do not need to check errors on actions for which error handling has already been defined.

## Version 7.1

- Internal updates

## Version 7.0

*MT Debugger for XPath Expressions and Actions*

- In the the XPath/XQuery Window[1244], you can not only build expressions using pop-up entry helpers, but also evaluate results and debug expressions.
- In the Actions Debugger[1390], you can debug actions[667]. Before or during a simulation[1355], you can select the actions you want to debug. The simulation will pause at these actions and display them in the Actions Debugger[1390].
- Set breakpoints and tracepoints on XPath expressions and breakpoints on actions[667]. Simulations can be paused at these breakpoints, and the expression or action is displayed in the respective debugger (XPath Debugger[1252] or Actions Debugger[1390]).
- A new Breakpoints Pane[269] in which debug points (breakpoints and tracepoints) can conveniently be managed in a single place.
- A new Debug menu[1626], which contains the commands relevant to debugging.

*Page properties*

- Length units of design components and text size in previous versions were given either as pixels or a percentage of a containing component. From version 7.0 onwards, page-related length units can be

also be specified as device-independent pixels (dp) and scale-independent pixels (sp) 1312. The built-in XPath extension function **`mt-convert-units`** enables you to convert between units.
- The *Browser Width* property has been renamed to *Browser Max Width* 384, and now specifies the maximum width of a solution's page in the browser.
- A page's properties 384 now enables page margins to be set.
- A new property named *All Styles* enables you to set all page styling properties 384 in one convenient location via an XPath map expression.

*Controls*

- Length units of design components and text size in previous versions were given either as pixels or a percentage of a containing component. From version 7.0 onwards, length units for control properties can be also be specified as device-independent pixels (dp) and scale-independent pixels (sp) 1312. The built-in XPath extension function **`mt-convert-units`** enables you to convert between units.
- The width of combo boxes 459 can be set (via the `wrap_content_longest_entry` value of the `Control Width` property) to be as wide as the longest item in the dropdown list of the combo box. Combo box widths can, as a result, be set more flexibly.
- Check Boxes 447 can be vertically aligned relative to its text if the text wraps over more than one line. This is done via the control's `Vertical Alignment` property.
- If a table is broader than its parent object, then its the width of any of its columns can be reduced 614 by using the new `wrap_content_fit_parent` value.
- With the `Skip wrap_content` 614 property of table cells, the content-width of a specific table cell can be disregarded for the `wrap_content` calculation (which determines the minimum width of the cell's parent column).
- Button control 417: Additional images have been added to the library of images that can be displayed on buttons (via the `Button Image` property).
- A new property named `All Styles` enables you to set all styling in a single property via an XPath map expression. For a description, see any control, for example the Button control 417.
- The handling of padding in table rows and columns 614 has been improved.
- Control templates can have their actions overridden for a particular instantiation 1208 by defining a new set of actions on the instantiating placeholder.
- The `Text Size Auto-Fit` property has been extended so that controls in control templates can be grouped to have a singe text-size. Nine such template groups can be defined. The property is available on controls 403 that display text, such as labels 554.

*Actions*

- The new Update Variable 903 action enables user-defined variables to be given new values during solution execution.
- The Scroll To 788 action has been extended to specify where, vertically in the view, the target object must scroll.
- The Measure Controls 927 action has been extended to take account of button background-colors and units specified as dp/sp lengths.
- The Print To action 686 provides an option to select the print output format via XPath, which enables the print format to be selected dynamically.
- The DB Begin Transaction 856 action provides a *Timeout (in seconds)* property for SQLite databases, which enables you to specify a wait period for applying a write-lock.
- The Update Display 790 action provides options for specifying which controls to update.
- The Template Event Callback 1208 action can be used on placeholder controls to modify the actions that the placeholder will execute.

*XPath-related features and XPath extension functions*

- The following new MobileTogether extension functions [1262] have been added: `mt-convert-units` and `mt-solution-path`.
- The MobileTogether extension function [1262] `mt-control-width` has been extended to take account of button background-colors and units specified as dp/sp lengths.

*Miscellaneous*

- The browser settings of projects [296] have been extended with two new settings: (i) a base size for calculating the font size of controls; (ii) whether back navigation in solutions that are embedded in IFrames is allowed or not.
- You can specify the default starting language of a multi-language solution when you deploy the solution [1574].
- User Variables [1309] can be defined to be stored on the client only, which improves speed when the variable contains or involves the use of large datasets.
- Support for Android 10.

## 2.4 Version 6

### Version 6.0

*Control templates and Placeholder Controls*

- A control template[1200] is a design component that is defined in one location and can be reused at multiple locations in the design. The parameters and variables of a control template provide flexibility, enabling use in different contexts.
- A new Placeholder Control[568] enables you to place a control template at a desired location. You can use multiple Placeholder Controls to place a control template at multiple locations.
- How to use control templates[1200] and Placeholder Controls[568] is described in the section Control Templates[1200].

*Controls*

- A new Placeholder Control[568] enables you to place a control template at a desired location and pass the template parameter values via XPath expressions.
- Button control[417]: Additional images have been added to the library of images that can be displayed on buttons (via the `Button Image` property).
- Button control[417]: A new property `Button Image/Text Distance` has been added to specify the horizontal distance between a button's image and text.
- Rich Text control[584]: The height of the control can be applied to web client display as well—in addition to its applicability on other devices
- The Open URL and Open File actions[681] on web clients have a new option that enable the resource to be opened in the current browser tab or a new browser tab.
- The Label control[554] has a new property, `Strikethrough Text`, which shows the label's text with a strikethrough.

*Actions*

- Action Execute FlowForce Job[838] enables the result of a FlowForce job to be returned as an action.
- The Measure Controls[927] action returns the minimum width in pixels of the specified control kind (button or label, for example) for a specified control text. This enables you to find the width of a set of controls and use this information in your design. For example, you could find the width of all buttons in a column and then set the width of this column according to the largest button-width value. The returned value is stored in the **`MT_MeasureControls`**[1304] variable.
- The Break Loop[899] action is placed inside a Loop action[897], and is used to exit the loop.
- The Open URL and Open File actions[681] on web clients have a new option that enable the resource to be opened in the current browser tab or a new browser tab.
- A new project option (in the More Project Settings dialog[296]) has been introduced to abort actions in case of errors during action handling.
- The Solution Execution[915] action has two new settings: (i) to restart the solution, and (ii) to enable solutions running on web clients to be opened in a new browser tab.

*Action Groups*

- You can now also pass arrays and maps to Action Groups as parameter values[943]. This is in addition to being able to pass atomic values and nodesets.
- In the Execute Action Group action[943], the Action Group to execute can be selected via an XPath expression, thus enabling the Action Group to be selected dynamically. This option is in addition to

being able to select the Action Group from a dropdown list that contains all the Action Groups defined in the design (a fixed selection).

- Action Group parameters 945 can also be selected dynamically, via XPath expressions.
- Action Groups 945 have been enhanced so that variables can be defined and used within the action group.

*JSON*

- Two new JSON-related MobileTogether extension functions 1262 have been added: (i) `mt-load-json-from-string`, (ii) `mt-save-json-to-string`. These enable you to, respectively, generate an XML node from a serialized JSON data structure, and generate a serialized JSON data structure from an XML node. They are useful when you want to pass JSON structures as string parameters, for instance, in a web services call.

*XPath-related features and XPath extension functions*

- The XPath/XQuery Window 1244 has been enhanced with improved entry helpers and more intuitive layout.
- Three new MobileTogether extension functions 1262 have been added: (i) `mt-db-original-row`, (ii) `mt-load-json-from-string`, (iii) `mt-save-json-to-string`.
- The **`MT_InputParameters`** 1300 variable now takes, by default, a map data structure. For individual projects, you can switch the data structure of this variable (in the More Project Settings dialog 296) to take a sequence of values (which was the mandatory data structure of the variable in releases prior to 6.0).
- The new **`MT_MeasureControls`** 1304 variable stores the result of the last-executed Measure Controls 927 action.
- The new **`MT_AutheticationToken`** 1300 variable stores the authentication token data sent by the Solution Execution 915 action.
- New Altova extension functions 1689 are available for use in XPath expressions. For descriptions of the currently available functions, see here 1689.

*Databases*

- A new MobileTogether extension function 1262 named `mt-db-original-row` retrieves the data from a row before the row was modified.
- When saving modified DB data, you can specify in what order the modifications must be made: Delete,Update, Insert. Such Save actions can be carried out via the Save control 803 and the Filter Columns command of the DB page source's context menu 359.

*Miscellaneous*

- If you want the user of a solution to go to a solution on another MobileTogether Server, then you can pass authentication information securely 1237 to the second server, thus obviating the need for a second user login.
- A new Java tab in the Options dialog 1678 to specify the location of the Java VM (virtual machine) on your system.
- Parameters and variables can be declared on a sub page 381. This provides you with more flexibility for dealing dynamically with data on the sub page.
- In the project settings 296, you can select the theme of the solution: light, dark, or device-selected.
- On Android 7 devices and newer, the app window has been configured for use in multi-window display (aka split-screen mode).

## 2.5 Version 5

Given below are the new-features lists of **Version 5** releases.

### Version 5.4

New features and updates in MobileTogether Designer **Version 5.4**:

- A Video Recording 1114 action starts the video recording app of client devices, and saves the recording to a location that you can define. Key properties of the recording can be specified.
- A Geolocation Map 508 control can be used to display the map of a specific area, in street, satellite, or hybrid view. Points of interest in the area can be shown by markers in the map. The control has a `OnGeoMapMarkerClicked` 508 event, for which actions can be defined.
- A MobileTogether extension function **`mt-geo-map-marker`** 1262 that creates a marker for the Geolocation Map control 508.
- A dynamic variable **`$MT_GeolocationMapMarker`** 1304, which contains information about the marker that was last clicked by the client's user. Information about the marker can be returned via an XPath expression.
- An Altova extension function **`geolocations-bounding-rectangle`** 1707 that creates a bounding rectangle around a set of submitted geolocations.
- The Save/Restore Page Sources action has been renamed to Backup/Restore Page Sources 806.

### Version 5.3

MobileTogether Designer **Version 5.3** adds the **`mt-is-server-purchased()`** 1262 function and an option, in simulations 1355, to simulate that the server has been licensed with purchased license/s.

### Version 5.2

MobileTogether Designer **Version 5.2** provides the ability to export a project (or design), together with its resources, as a MobileTogether Package 295. The package is saved as a `.mtp` file, which can be opened in MobileTogether Designer and deployed as a solution (together with resources) to MobileTogether Server.

### Version 5.1

New features and updates in MobileTogether Designer **Version 5.1**:

*Controls*

- Two useful formatting options for Tables have been added: (i) The `Background Color` property of a table cell can take a value that sets the cell's background color to be the same as that of the control that is in the cell; (ii) The table property `Apply Borders to Cells` automatically passes the table's border properties to the borders of all cells in the table.
- If the text of a Label 554 contains URLs (such as `www.altova.com`) or email addresses (such as `altova.user@altova.com`), then such text can be set, via the `Automatic Link Detection` property, to be automatically displayed as a live link in the label text.
- A maximum number of lines can be specified for Label controls 554 that have been set to multiline display.
- The auto-sizing of text applies not only to single-line text in controls, but also to text in controls 403 that have been set to multiline display.

*Actions*

- The Read Folder[847] action has been extended so that: (i) multiple filename patterns can be specified in the action's file filter; (ii) an option to recurse into subfolders is provided; (iii) if recursing into subfolders is specified, an option to read the information of empty folders is provided.
- The Send SMS To[698] and Make Call To[679] actions can be started directly in designs that have been generated as AppStore Apps[1471], but need the end user's permission in designs that are deployed as MobileTogether solutions.
- In addition to being able to save entire page sources to file, you can optionally save sub-trees of page sources[809] to separate files.
- The View Image[713] action has been enhanced with a fit-to-screen option.
- The `OnPageRefresh`[390] event (of a page) has a new option: *Refresh due to Orientation or Size Change*. This option specifies the actions to carry out when the end user changes the device's orientation or re-sizes the app window (on devices where the window is re-sizable).

*Miscellaneous*

- Android support has been extended to Android 8.1 and 9.
- Support for round and adaptive icons in appstore apps[1472] for newer Android versions.
- When generating code for appstore apps[1472], you can view image files of launcher icons directly from the code generation dialog.
- In the project's settings[296], you can specify that the device settings be saved with the design.
- XQuery page sources can be specified to be persistent on the client[359], via the XQuery tree's context menu[359].
- When a solution is opened in a web client, client files are stored at a session-specific location on the server and are deleted from this location when the session ends. See, for example, information about file locations of the Save File actions[799] of page sources.
- Ability to disable, for individual pages of a project, the message box that asks the end user to confirm their wanting to leave the page. The message box is set for all pages of the project in the Browser Settings of the project[296], and it is disabled for individual pages by assigning a value of **`mt-no-browser-exit-confirmation`** to the Browser CSS Class property of the individual pages[384].
- A new XPath extension function for MobileTogether[1262], **`mt-run-web-url`**, to generate a URL that opens a specified solution in a web browser.
- The Simulator window[1356] provides the ability to search in the Page Sources pane for text in page source nodes and data.

## Version 5.0

New features and updates in MobileTogether Designer **Version 5.0** are listed below.

*Controls*

- New border properties[614] (width, color, and style) have been introduced for table items[614] (cells, columns, rows, and table).
- Border settings can be applied quickly to different table items by using the Border Settings dialog[1644].
- Shorthand `margin` and `padding` properties have been added to controls that have margin and padding properties. This enables a common value for all four sides to be specified in a single property, rather than individually for each side.

- Padding properties have been added to more controls[403] where padding is applicable. This enables space to be added between individual borders of a control (left, right, top, and bottom) and the control's content.
- In a Combo Box[459] control that allows multiple values to be selected by the end user, different separators can be used to construct (i) the string that is entered in the page source node and (ii) the string that is displayed in the combo box. The former is set via the `Multi Select Separator` property, the latter via the `Multi Select Separator Visible` property.
- In Combo Boxes[459],, the text of dropdown list items and the corresponding XML data values can be defined via an XPath expression that returns a sequence of two-member arrays. This way of defining combo box entries is in addition to existing options.
- The appearance of Buttons[417] has been enhanced so that both a Button image (see the `Button Image` property) as well as a Button text can be displayed. For the Button image, you can choose from a range of predefined icons or use a custom image. You can also set the position of the Button image to be to the left or the right of the Button text.
- The **`Text Size Auto-Fit`** property of Controls that have a Text Size property[403] has an additional option. Text that is too long to fit within the width of a control is shown as a truncated string that ends with an ellipsis. This is in addition to existing options for auto-fitting text.
- Values for the `Checked Values (true/false)` property of the Check Box[447] and Switch[603] controls can also be set in style sheets[1318]—which enables this property's values to be set globally for these controls.
- The `Checked Values` property of the Radio Button[571] control can also be set in style sheets[1318]. This enables this property's checked value to be set globally.
- The *OnEnter/OnEscape* events of the Button[417], Chart[438], Image[541], and Label[554] controls have been enhanced for use on all client devices (in addition to their use on Web and Windows clients).
- The Rich Text[584] control can use a predefined Rich Text style sheet[1227] for text that is marked up with HTML tags.
- The text that will be the content of the control[584] can be defined with an XPath expression that evaluates to an HTML-encoded string.
- The Vertical Line[651] control can be given top and bottom margins.
- The Horizontal Slider[520] control has a `Auto Correct Value` property that automatically corrects values in the associated page source node to a value that is within the defined slider value range.
- A context menu command for controls in the Controls Pane[266] displays all instances of that control type.

*Actions*

- A new Quick Filter text box in the Actions dialog[668] enables you to filter the actions and action groups of the dialog.
- Actions and action groups can be added to the actions of an event from a popup in the event pane[668]. This is in addition to the standard way of adding an action by dragging it into the event pane.
- A new action, Load/Save Text File[821], enables (i) text to be loaded from a file to a page source node, and (ii) for text to be saved from a page source node to a text file.
- A new action, DB Read Structure[867], enables the structure of a DB to be read and for data in the DB to be stored in a new type of page source, the `$MT_DBSTRUCTURE` page source. Data in this page source can be used in te same way as data in any other page source. This action is useful if you want only a data read-out.
- The Open URL/File[681] action has been enhanced to accept data URLs. As a result, binary files can be opened directly in a new tab of the web client's browser.
- The Update Node(s)[886] action enables multiple nodes to be updated by specifying the target nodes in an XPath array.
- The text color of Comment[923] actions can be customized.
- Where an action's settings includes the selection of a page source, this selection can additionally be specified via an XPath expression.

### *Rich Text*

- The Markup toolbar icon[1233] of the Rich Text control in the deployed solution has been enhanced to let the end user select from a range of markup-tag sizes.
- You can specify, in the Browser Settings dialog[296], the fonts that are available to the end user when editing rich text[1233].

### *Special MobileTogether-related XPath extension functions*

- Six new MobileTogether extension functions[1262] have been added: (i) `mt-available-db-connection-names`, (ii) `mt-called-by-enter-key`, (iii) `mt-called-by-escape-key`, (iv) `mt-get-page-source-structure`, (v) `mt-table-rowgroup-count`, (vi) `mt-table-rowgroup-index`.
- A number of new Altova extension functions[1689] are available for use in XPath expressions. For descriptions of the currently available functions, see here[1689].

### *Databases*

- A new action, DB Read Structure[867], enables the structure of a DB to be read and for data in the DB to be stored in a new type of page source[349], the `$MT_DBSTRUCTURE` page source. Data in this page source can be used in te same way as data in any other page source. This action is useful if you want only a data read-out.
- The Simulation 2 tab of the Options dialog[1669] provides a new option to generate DB connections to an XML file, which can be used for simulations of the DB Read Structure[867] action.
- A new MobileTogether extension function[1262] named `mt-available-db-connection-names` gets the names of all available DB connections in the solution or on the server.
- The new **Replace DB Sources**[1613] command enables you to switch the DB connection of DB page sources in the design to alternative DBs. One use case would be to test with a non-production DB, and then switch to a production DB when the solution is deployed.

### *Miscellaneous*

- A new Find & Replace Pane[283] enables you to search for strings in the design, including in XPath expressions, functions, and action groups. Found strings can also be replaced.
- A new Listings Pane[281] displays various kinds of lists; for example, lists of all global variables, all user-defined functions, or instances of a particular control type. These listings contain links to the relevant design components, which enables you to quickly find a design component and go to it.
- You can copy a pre-existing style sheet and paste it[1318] as a new style sheet.
- MobileTogether Server Services[1543] can be configured to send push notifications (PNs)[753] to standard MobileTogether solutions as well as to MobileTogether AppStore Apps[1471].
- The Network Proxy options dialog[1676] enables you to configure custom proxy settings for the application.
- The SPL language[1492] that is used to generate program code for AppStore Apps for Android, iOS, and Windows[1471] has been extended with functionality to trim strings of specified characters from left and right[1495].
- The context menu of the Styles & Properties Pane[274] has been extended with commands to (i) list controls having the same value as that of the selected property, and (ii) display controls in groups according to the values of a given property.
- Page sources that are supplied with data from a FlowForce job[317] can now accept HTML or JSON data, as alternatives to the already supported XML format.
- REST functionality[328] in MobileTogether now supports the sending of files; this is in addition to the sending of XML and Base64 data.

- Multiple font files can be embedded in the solution. These files can be referenced via CSS when the solution is displayed in web clients. The font files to embed are specified in the Browser Settings of the project's properties[296].

# 2.6 Version 4

Given below are the new-features lists of **Version 4** releases.

## Version 4.1

New features and updates in MobileTogether Designer **Version 4.1**:

*Server services*

- A server service is a set of MobileTogether Designer actions that is deployed to MobileTogether Server **Advanced Edition** as a solution (`.mtd` file). The service is executed on the server when a specified set of MobileTogether Server conditions is met. (These server conditions are defined in the administrator interface of MobileTogether Server Advanced Edition.)
- A server service is defined in a server service design, which is opened via the **File | New Service** 1563 of MobileTogether Designer.
- How to create a server service in MobileTogether Designer is described in the section Server Services 1543.
- A `$MT_SERVICE` 1545 page source is automatically created when a service design is created. It holds run time data about service triggers.
- The **`$MT_SERVICE`** page source can be manually filled to simulate run time data about service triggers 1383.

*Rich Text*

- A new Rich Text control 584 enables text from a page source to be displayed with formatting (on all clients) and edited (on Windows and Web clients). The formatting can be based on styling markup in the XML page source or can be added by you. In both cases, the rules are specified in a Rich Text style sheet 1228.
- For each project (design), you can define multiple Rich Text style sheets via the Rich Text Style Sheets dialog 1598. Any one of these style sheets can be assigned to a Rich Text control 584 so that the text displayed in the control is formatted according to the rules of the selected style sheet.
- For an overview and description of this feature, see the section Rich Text 1225.

*Actions*

- The Go to Subpage 783 action has been enhanced with an option to open the subpage as a modal dialog (that is, in a separate window above the current page). This is an alternative display to that of replacing the current page with the subpage.
- The Save/Restore Page Sources 806 action enables you to save a page source provisionally, and then to accept or discard further modifications on the basis of whether one or more conditions are met.
- The Access Calendar 673 action saves information about the device's calendars and calendar events to the `$MT_CALENDAR` page source 349. It also enables events to be written to a calendar on the device. For simulations, the calendar of Microsoft Outlook 1663 or an XML file can be used.
- A Replace Node(s) 883 action provides a mechanism to delete nodes from the node of a page source, and to then add new nodes to it.

*Controls*

- The Combo Box control 459 is enhanced to enable users to select multiple options (via the control's **`Multi Select`** property).

- Controls that have a Text Size property 403 now additionally have a **`Text Size Auto-Fit`** property, which enables text to be automatically re-sized to fit the width of the control. Controls can also be assigned to a group, so that all controls have an automatically selected uniform and reasonable size. All the controls of a page for which the auto-fit property has been set can be listed in the Listings pane by using the Page menu command **List Text Size Auto-Fit Groups** 1636.
- In a design, some controls can be assigned to a "tab order sequence". When an end user then clicks the **Tab** key repeatedly (on Web and Windows clients), the solution's focus will move through the controls in the specified tab order. The entire tab order can be set via the **Page | Show/Define Tab Order** 1633 menu command. The position in the sequence of individual controls can also be set in the **`Tab Order`** property of the control. The controls that can be assigned positions in the tab order sequence are: Buttons 417, Check Boxes 447, Combo Boxes 459, Dates 473, Edit Fields 495, Radio Buttons 571, Switch controls 603, Time controls 640.
- Controls that have an **`OnClicked`** event (Buttons 417, Charts 438, Images 541, and Labels 554) can have their click events triggered via the client's **Enter** or **Escape** key (on Web and Windows clients). The setting for this can be made via the `On Enter/Escape` property of the control or in the dialog for defining the control's **`OnClicked`** event actions. See the description of the respective control.

*XPath extension functions*

- Two new MobileTogether XPath extension functions 1262: (i) **`mt-client-ip-address`** (to obtain the device's IP adddress); (ii) **`mt-image-width-and-height`** (to get the dimensions of the submitted Base64-encoded image).
- A new Altova XPath extension 1764 **`generate-guid`** generates a unique GUID string that can be used as an id.

*Miscellaneous*

- Enforce Light Theme 296: In the Project Properties 296 pane, you can specify whether the pages of the project must have a light background (dark text on light background) or not. The default value of `false` specifies that the client-specific theme will be used.
- The contacts manager and calendar of Microsoft Outlook can be used for simulations of the Read Contacts 692 and Access Calendar 673 actions. This is done by selecting the corresponding items in the Options dialog 1663.

## Version 4.0

New features and updates in MobileTogether Designer **Version 4.0** are listed below.

*Push Notifications*

- A push notification (PN) is a text message that is sent from one solution to a mobile device on which a receiving MobileTogether solution is installed. When a PN is received, it triggers a set of actions in the receiving solution. For an overview of the PN feature, see the section Push Notifications 1125.
- The Send Push Notification 753 action is specified in the sending solution. It defines the various parameters of the PN that is to be sent.
- In the receiving solution, actions for the **`OnPushNotificationReceived`** 296 event specify what actions are to be carried out when a PN is received.
- Besides a text message, the PN also carries a payload. The payload is automatically transferred to the `$MT_PUSHNOTIFICATION` page source of the receiving solution.
- A PN can contain buttons. PN buttons are specified in the Send Push Notification 753 action of the sending solution. While definitions of the buttons for non-iOS devices are made directly in the Send

Push Notification[753] action, for iOS devices, the buttons are defined in the receiving solution by using the **Project | iOS Push Notification Button Sets**[1608] command.

- An external PN key[1128] is a text string that is used to identify a mobile device. The Register Ext PN-Key[757] action associates a mobile device with a string that you specify. An external PN key is used to identify a set of mobile devices that will receive a PN. A reverse action, Unregister Ext PN-Key[757], is also available.
- A PN topic is a text string that names a topic. The Register PN-Topics[759] action associates a mobile device with one or more PN topics. If a PN is sent to a PN topic, then all devices that have been associated with that topic will receive that PN. A reverse action, Unregister PN-Topics[759], is also available.
- If a PN is sent to a different receiving solution, then, for simulations of the receiving solution to be successful, the incoming PN must be simulated. A mechanism to simulate incoming PNs is available in teh simulator. It is described in the section Simulating Push Notifications[1133].
- A MobileTogether solution that uses PNs can be compiled into an AppStore App[1471]. A few additional steps are required to compile AppStore Apps[1471]. These steps are described in Push Notifications in AppStore Apps[1130].

*Embedded Webpage Solutions*

- A new Embedded Webpage Solutions[1427] feature that enables solutions to be embedded inside webpages by way of IFrames. Data can be exchanged between the webpage and its embedded solution. The solution itself interacts with MobileTogether Server as usual and receives data that can then be communicated back to the webpage. Authentication via JSON Web Tokens (JWT)[1439] enables embedded webpage solutions to be integrated into existing systems.
- The **`OnEmbeddedMessage`**[397] event is triggered when a solution's workflow on the server receives a message from the embedded solution.
- The **`$MT_EMBEDDEDMESSAGE`**[349] JSON page source (structure and data) is created when the **`OnEmbeddedMessage`**[397] event is triggered.
- The Load from String[829] action parses a string and generates a (JSON/XML) page source from it.
- The Save to String[829] action serializes a selected (JSON/XML) page source, and saves the serialized string to a specified location.
- The Embedded Message Back[924] action sends a serialized JSON string as a `message` event to the IFrame that loaded the current solution.

*New actions*

- The MapForce Transfer[839] action supplies a MapForce Server Execution file (MFX file) to MapForce Server for processing. A set of input data structures can, in this way, be transformed into a new set of data structures (the output of MapForce Server). This enables legacy data structures—or other data structures that cannot be modified—to be used in a MobileTogether design.
- The Read Folder[847] action reads the contents of a specified folder and passes metadata about each folder item to a separate node of the `$MT_FILEINFO` page source.
- The Set Language[928] action enables the language of the solution to be changed by the user. This enables a solution to be restarted in an alternative language when a specific event is triggered.
- The Load from String[829] action parses a string and generates a (JSON/XML) page source from it.
- The Save to String[829] action serializes a selected (JSON/XML) page source, and saves the serialized string to a specified location.
- The Embedded Message Back[924] action sends a serialized JSON string as a `message` event to the IFrame that loaded the current solution.
- The Send Push Notification[753] action defines the various parameters of the push notification that is to be sent.

- The (Un)Register Ext PN-Key 757 action registers a text string as the external Push Notification key of a solution on that mobile device. See the section Push Notifications 1125 for more information.
- The (Un)Register PN-Topics 759 action registers a device to receive Push Notifications about one or more selected topics. See the section Push Notifications 1125 for more information.

*Miscellaneous*

- The MobileTogether Server installation is pre-deployed with a powerful solution that displays access statistics about individual solutions on that server; for example, the frequency of access, and the number of devices and type of device accessing a particular solution. For more information about the **`Statistics`** solution, see the MobileTogether Server documentation.
- User-defined tools can be created in the Tools tab of the Customize dialog 1658. The tools created in this way are accessed via commands in the **Tools | User-Defined Tools** 1655 menu.
- A new Table menu 1637 that provides table-related command in one menu to help you quickly design and edit table structures.
- The new command **List Usages of All Style Sheets** 1612 displays all the style sheets defined in the project (including unused style sheets), and the page, table, and control instances that use these style sheets. Unused style sheets are also shown in the list generated when the **List Unused Functions, User Variables, Style Sheets, and Action Groups** 1613 command is clicked.
- Users can swipe right/left to horizontally scroll tables 1078 that are broader than the viewport.
- The simulator's menu 1355 provides options to simulate the availability of the following mobile device functionality: (i) the camera app, (ii) the gallery, (iii) the microphone, (iv) NFC 1118, (v) GPS location, (vi) the address book, (vii) telephony services, (viii) SMS services. With these options, design scenarios can be tested that require these services to be available on the device.
- Log messages (shown in the Messages Pane 278) that relate to specific actions of specific events can be suppressed or enabled as needed 667.
- Page source data can be automatically reset when the solution exits a page. This is done with the **Reset Data** 359 command, which is available in the context menu of page sources.
- When saving (any type of) files 809, optionally, a default file extension can be specified; this extension will be used if none is specified with the file name.
- During simulations 1355, you can copy the XPath locator expression of any page source node to the clipboard.

# 2.7 Version 3

Given below are the new-features lists of **Version 3** releases.

## Version 3.2

New features and updates in MobileTogether Designer **Version 3.2** are listed below.

*Near Field Communication (NFC)*

- A new NFC feature[1118] for sending and receiving messages via NFC. Additionally, on Android devices, Android Beam can be used to send files. For an overview of all the design components that are used to implement this feature, see Design Components for NFC[1123].
- NFC-related events[1121] to trigger actions: `OnPushNdefMessageCompleted` and `OnNFCTagDiscovered`.
- A new MobileTogether extension function[1262] to check whether NFC has been started: `mt-nfc-started`. Plus functions to convert text and Base64 to/from hexBinary[1123] (since NFC message payloads are encoded in hexBinary).
- NFC sample files[1377] enable the simulation of NFC-tag discovery.

*Text to Speech*

- A new Text to Speech feature[1111], based on the Text to Speech action[727], enables text strings to be converted to speech and played back.
- New MobileTogether extension functions[1262] for providing information related to the Text to Speech feature[1111]: `mt-text-to-speech-is-language-available` and `mt-text-to-speech-is-speaking`.

*Miscellaneous new actions*

- A new Read Contacts[692] action to store the contacts of the device's address book in a data source tree.
- A new Get File Info action[849] to store the file information of a specified file (such as size, creation date, etc) in a data source tree.
- A new Wait Cursor[702] enables a platform-dependent wait cursor to be displayed while an action is being executed; this is useful for actions that require a long time to complete.
- New Let User Choose Date[677] and Let User Choose Time[678] actions enable the end user's selection of date and time, respectively, to be saved to page source nodes.
- A View Image[713] action enables an image from the client device, or a data source node, or an image/chart/signature control to be displayed.
- A Try/Catch Server Connection Errors[908] action can be used to try for exceptions on specific server transactions. You can define appropriate actions to take should a connection error occur.

*Enhancements of existing actions*

- A new command to show all usages of an action or action group[667] in the design.
- The Audio action[720] has been enhanced to enable the playback of predefined sounds available on the client device. Currently, you can select from among 16 predefined sounds.
- The Send Email (via server) action[693] now contains a *Reply To* setting. This enables emails sent via MobileTogether Server to have both "pseudo" and real sender-addresses.
- The Reset action[802] now enables all data sources, including the `$PERSISTENT`[349] tree (for persistent data on the client), to be reset.
- The Show Geolocation[740] action additionally accepts an address for the geolocation to show on the map app of the client device. Previously only latitude/longitude coordinates were accepted.
- The Scroll To[788] action replaces the Scroll to Bottom action of previous releases. The new function has been enhanced to additionally enable scrolling to a specified control or to the top or bottom of a

specified table. If you have used the older Scroll to Bottom action in a design and open that design in this (or a later) version of MobileTogether Designer, the action will be automatically translated into the new action.
- A new MobileTogether extension function[1262] to check whether geolocation tracking has been started: `mt-geolocation-started`.
- The target pages of the Go to Page[783] and Go to Subpage[783] actions can additionally be set via XPath expressions.
- In message boxes containing custom buttons[679], you can specify actions that will be performed when the device's **Back** button is tapped.
- The Try/Catch expression of previous releases has been renamed to Try/Catch Exceptions[907].

*New features of tables*
- Tables can have dynamic columns[1074], meaning that columns can be added dynamically on the right-hand side of the table according to the number of instances of the element that corresponds to the column-field in the design.
- A dynamic, local variable[1304], **`MT_TableColumnContext`**, has been added. It provides the context node of the current column during the generation of tables. See the section Dynamic Columns[1074] for a description of usage.
- The number of rows that can be loaded in scrollable tables[1085] can be set with the **`Row Group Chunk Size`**[614] table property.

*Enhancements for controls*
- Two new control properties[404] are available for controls that can be enabled/disabled: `Text Color (Disabled)` and `Background Color (Disabled)`. These enable different colors to be set for a control depending on its state (enabled or disabled).
- Additional Button looks[417] are available: *Import, Export, Calendar,* and *Time*.
- Images that have been embedded[541] in the design file as Base64 data can be quickly re-embedded, that is, re-converted from binary to Base64 and stored in the design. This is done via the image control's context menu. This feature facilitates the updating of an embedded image file if the image has been modified.

*Miscellaneous*
- In the Pages Pane[257], you can check for references (in the design) to a page by selecting the context menu command **List Usages in Actions**.
- Text that is copied from the Edit XPath Expression dialog[1244] can be pasted as XPath into the Styles & Properties Pane.
- Additional toolbar icons in the Style Sheets dialog[1318] for controlling the display of items: (i) expand all items; (ii) collapse all items; (iii) display non-empty items only.

## Version 3.0

New features and updates in MobileTogether Designer **Version 3.0** are listed below.

- The Style Sheets feature[1318] enables you to define global styles that can be applied at the project, page, table, and control level. This provides a one-stop repository of cascading styles for the project.
- The Print To[686] action uses Altova StyleVision Server to generate PDF, Word, and RTF documents from XML data.
- The Open URL action has been enhanced so that it is now the Open URL/File[681] action. Previously, this action opened Web pages in the browser of the client device. The action now enables files on the client device to be opened in the device's default application for that filetype.
- The Let User Scan Barcode[714] action opens the client's camera application and enables users to scan a barcode; the barcode data is entered into an XML data tree and can be processed further.

- Two new properties provide better layout control: (i) the project property **`Top-Level Margins`**[296] (available via More Project Settings in the project properties that you can set in the Styles & Properties entry helper); it enables margins to be set for all top-level controls of a page; this essentially sets a margin for the page; (ii) the table property **`Table Padding`** [614]switches the padding of tables on iOS devices on or off.
- The Automated Testing[1399] feature enables you to compare two test runs to detect differences in the design, page source data, and the solution environment.

# 2.8 Version 2

Given below are the new-features lists of **Version 2** releases.

## Version 2.2

New features and updates in MobileTogether Designer **Version 2.2** are listed below.

- An option to enable the end user to select, on the client device, the client file to load/save is available for the following actions: Load/Save File[809], Load/Save Image[707], and Load/Save Binary File[815].
- Video controls[655] enable videos to be played on a page. The control's properties and video-related MobileTogether extension functions[1262] enable the playback and the control to be customized. For an overview of video features, see the section Audio, Video[1108].
- A Video action[728] enables videos to be started, paused, resumed, stopped, and searched. Playback of specific time-defined segments can also be defined. For an overview of video features, see the section Audio, Video[1108].
- An Audio action[720] enables audio files to be started, paused, resumed, stopped, and searched on five audio channels. You can also select specific time-defined segments for the playback. For an overview of audio features, see the section Audio, Video[1108].
- The **`$MT_AudioChannel`** global variable[1304] gives the number of the audio channel that triggered the action[1109].
- An Audio Recording action[724] enables audio to be recorded to a file on the client device. For an overview of audio features, see the section Audio, Video[1108].
- New MobileTogether extension functions[1262] for providing information about audio and video files, and about actions related to audio and video: `mt-audio-get-current-position`, `mt-audio-get-duration`, `mt-audio-is-playing`, `mt-audio-is-recording`, `mt-video-get-current-position`, `mt-video-get-duration`, `mt-video-height`, `mt-audio-is-playing`, and `mt-video-width`.
- New MobileTogether extension functions[1262] for providing information about the last client file that was used: `mt-last-file-path`, `mt-extract-file-extension`, and `mt-extract-file-name`.
- New button icons related to the audio/video features can be selected via the `Button Look`[417] property.
- New global variables[1304] **`$MT_WindowHeight`** and **`$MTWindowWidth`** dynamically give the dimensions of resizable browser windows and of app windows on Windows systems.
- A Load/Save Binary File action[815] enables: (i) any type of binary file to be loaded into the solution as Base64-encoded XML content, and (ii) Base64-encoded XML content to be saved as a binary file.
- The Send Email To[693] action can send text-file attachments, in addition to XML files and binary files.
- The Simulator[1356] can be set to simulate the availability of a LAN connection. This adds to the number of connection types[1663] that can be simulated, which now are: mobile network, WiFi, and LAN. A related MobileTogether extension function[1262] has been introduced: (i) `mt-connected-via-lan`.
- Table headers and footers can be added to dynamic tables via the context menus of tables[1087].

## Version 2.1

New features and updates in MobileTogether Designer **Version 2.1** are listed below.

- Data files, such as XML[355] and image[707] files, can be loaded from client devices and saved to client devices.
- Two new controls are available: Vertical Line[651] and Horizontal Slider[520].
- The following actions have been introduced:
  - Share[699]
  - Cancel Action Execution[913]
  - User Cancel Behavior[917]

  - Restart/Stop Page Timer[791]
  - Delete File/Folder[853]
  - DB Bulk Insert Into[865]
  - Let[900]
  - Try/Catch[907]
  - Throw[905]
  - Return[909]
- Action Group Results[950]: A Return action[909] in an Action Group generates an Action Group Result, which can be used as the value of a variable defined in a Let action[900].
- Action Groups can take parameters[943]. Additionally, an Action Group itself can be set as as the value of a parameter[945].
- The Close Subpage action[788] has been extended to return a value that can be used as the value of a variable defined in a Let action[900].
- The Show Geolocation (on map)[740] action has been enhanced to show the routes between two locations.
- Emails sent from clients[693] can be in HTML format.
- The precision of timers[390] used for the page refresh event[390] has been increased to milliseconds.
- XPath definitions of the following properties: `Keyboard` (of the Edit Field control[495]), `Horizontal Alignment`, and `Vertical Alignment`.
- The width of controls[404] and table columns[614] can be set in pixels.
- Tables: A whole table or a part of a table can be designed to be scrollable[1085]. Scrollable tables[1085] can be specified to fill the screen height.
- Tables: Separate visibility settings for spanned columns and rows[1078].
- Tables: Background colors can be assigned to individual rows and columns[614] (in addition to cells).
- Tables: Nested tables can be assigned horizontal-alignment and vertical-alignment property values[614].
- The `Keyboard` property of the Edit Field control[495] has been enhanced with the *Visible Password* value. As a result, you can define whether to hide or show passwords when the end user types one into an edit field.
- The Button control[417] has additional predefined looks (specified via the `Button Look` property), including transparent buttons.
- New MobileTogether extension functions[1262]: (i) `mt-connected-via-wifi`, (ii) `mt-control-width`, (iii) `mt-font-height`. Font sizes, in pixels, can be generated with XPath expressions that use the [1262]**mt-font-height**[1262] function[1262].
- When saving to a DB, columns can be filtered separately[1035] depending on whether data is being updated or inserted.
- The `Show Page Title Bar`[384] page property enables the page's title bar to displayed or hidden.
- User-generated AppStore Apps[1471]: The app's UI language[1472] can be selected from among English, German, French, Spanish, and Japanese.
- Duplication of custom localization strings[1600].

## Version 2.0

New features and updates in MobileTogether Designer **Version 2.0** are listed below.

- Designers can create their own MobileTogether custom apps that end users can download to mobile devices. We call these apps AppStore Apps. The section AppStore Apps[1471] describes how to generate the program code for such apps from MobileTogether Designer. Code can be generated for Android, iOS, Windows (touch-enabled devices and PCs), and Windows Phone. After the code has been generated, it can be compiled into the corresponding AppStore App.
- Solutions on mobile devices can be suspended (paused and minimized). A new project property, *On Switch to Other Solution*[296], can be set to suspend the solution when the end-user switches to another solution. The end-user can switch back to the minimized solution by clicking its icon in the

*Running* tab of MobileTogether Client. Another way to specify whether a solution is canceled or suspended is via the Solution Execution 915 action.

- A Signature Field 590 control enables end-user signatures to be stored as images in a data source node.
- You can define and test actions to take when server connection errors 394 occur.
- Simulations have been enhanced 1356 to better emulate actions defined in the design. For example, server connection errors are simulated by an option to prevent server access 394.
- JSON data sources 317 can be used as page sources.
- Page data can be accessed and saved via REST requests 328. Such data can be used in page sources 317, and can also be accessed or saved via page source actions 799.
- REST requests support OAuth authorization 328. Each design has a pool of settings that can be used anywhere in the document. The settings can be managed in the Maintain OAuth Settings 1606 dialog. Furthermore, settings can be imported 1607 into the active document from other open MobileTogether Designer documents.
- Page data can be accessed and saved via SOAP requests 337. Such data can be used in page sources 317 and page source actions 799.
- New actions: Execute SOAP Request 835, Execute REST Request 837.
- The data retention option for page sources 345 offers considerable flexibility about whether data is stored on the client or server.
- A page event, `OnServerConnectionError` 389, has been added.
- Two dynamic, local variable 1304s have been added: **`MT_HTTPExecute_Result and`** **`MT_ServerConnectionErrorLocation`**.
- Commands to list all files, directories, and external page sources 1589 that are used in the project.
- Cells of Repeating Tables 1065 and Dynamic Tables 1069 are associated with page source nodes via XPath expressions, and were previously read-only. The content of such cells are now editable.

## 2.9 Version 1

Given below are the new-features lists of **Version 1** releases.

### Version 1.5

New features and updates in MobileTogether Designer **Version 1.5** are listed below.

- A Send Email To 693 action enables emails to be sent during the execution of a solution.
- The MobileTogether extension function **`mt-email-attachment`** 1262 creates text and image attachments for emails that are sent with the Send Email To 693 action.
- Links can be placed in the body of emails 693 that are sent as HTML. These links can target Internet pages and MobileTogether solutions.
- The control events and page events of a solution can trigger links that go to other MobileTogether solutions. Furthermore, the URLs that point to the MobileTogether solutions can contain URL query strings, which allow specific page contents to be displayed. See Hyperlinking to Solutions 1239.
- Hyperlinks that target solutions pass their URL query parameters to the targeted solution. These parameters can be stored in the **`$MT_InputParameters`** 1300 global variable, from where they can be referenced.
- Three link-related MobileTogether extension functions have been added: **`mt-run-solution-url`** 1262, **`mt-run-solution-url-parameters`** 1262**, and** **`mt-html-anchor`** 1262.
- A powerful Loop 897 action enables reiteration over a set of nodes, and thereby provides more design possibilities and solution functionality.
- Two other actions have been introduced: Hide Keyboard 790, and Update Display 790.
- A new radio button 571 control has been introduced.
- The strings of a solution automatically appear in the language of a mobile device if the solution has been localized 308 in that language. In this release, the default and localization strings can be exported/imported between the project and separate XML files 1600 for each language. This enables individual translators to work independently of each other translating the default-language strings into their different target languages. Each translated XML file can be imported separately back into the project.
- When entering the **`mt-load-string`** 1262 function in an XPath expression in the Edit XPath/XQuery Expression dialog 1244, all the custom strings 1600 defined in the project are displayed in a popup. The value of the string in the simulation language 1606 currently selected in MobileTogether Designer is also displayed.
- A new function, **`mt-localized-string-name`** 1262, returns the control name or string name of the submitted (localized) string.
- The button 417 control has the new `Button Look` property that enables an icon to be added as the button display from a predefined selection of icons.
- The horizontal line 515 control has the following new properties: `Line Style`, `Margin Top`, `Margin Bottom`.
- The width of all controls 404 can be specified as a percentage of the page width (via the control's `ControlWidth` property).
- Click events have been differentiated according to how long the user clicks the control. Taps on the control are `On Click` events, while longer presses are `On Long Click` events. Click events are available for the following controls: Buttons 417, Charts 438, Images 541, and Labels 554.
- The Insert Node(s) 880 and Append Node(s) 875 actions have an option to remove the inserted/appended node/s from their original locations in a project's page sources.
- Keyboard shortcuts for adding actions 667 to the definition of an event.
- Each control in the design 404 can have one or more class names assigned to it via its `Browser CSS Class` property. Rules for class selectors can be defined in an external CSS file, which must be

deployed to the server. The reference to this external CSS file is defined in the project's browser settings 296.
- An external CSS file 296 can be used to store additional CSS styles.
- A new dialog for a project's browser settings 296 collects the settings that define the behavior of the browser in the client mobile device.
- Custom fonts 296 can be embedded in a design.
- Enhancements to the XPath/XQuery Expression dialog 1245 include interactive functions-and-operators information in popups, information about global variables 1298 and custom strings 1600.
- User-Defined XPath/XQuery Functions 1293 can be ordered in the ascending/descending/dialog order of function names.
- Updating server settings on client devices 291.

## Version 1.4

New features and updates in MobileTogether Designer **Version 1.4** are listed below.

- Support for geolocation retrieval and processing, which is a vital feature for transportation-based mobile solutions. Actions to track, read, and display geolocation data 733 can be defined for events. Additionally, Altova XPath extension functions for manipulating geolocation data 1707 can be used in the design's XPath expressions. Geolocations can also be set for designer and server simulations 1372 so that geolocation input can be tested in the simulator.
- Support for XQuery 3.1, which provides new features for using maps, arrays, data in the JSON format, and more. You can use the Edit XPath/XQuery Expression Dialog 1244 to create and check XQuery expressions.
- String localization 308 (translation into additional languages) enables translations of the strings of a solution to be stored with a project. The language in which the solution runs is automatically selected to be the same as that of the mobile device. You can test localized solutions by running simulations in a specific language 1606.
- Specific headers can now be added to HTTP requests 317. This is in addition to parameters that can be specified in the HTTP request.
- Solutions can be chained to execute one after the other. The next solution to execute is specified in an option of the Cancel Solution action. [The Cancel Solution action is obsolete since v2.0; it is superseded by the Solution Execution 915 action.]
- Simulations 1355 have been enhanced for iOS7/8 rendering and for XML tree editing 1356. Being able to modify the XML tree in the simulator and see the resulting changes immediately in the simulation speeds up testing.
- The Project menu 1589 contains commands to show: (i) used global and page source variables 1610; (ii) used user-defined XPath/XQuery functions 1611; (iii) used action groups 1611; (iv) as well as unused variables, functions, and action groups 1613. This improves the maintenance and development of large, complex solutions.

# 3 Introduction

This section provides an overview of MobileTogether and MobileTogether Designer. It contains the following sections:

- MobileTogether Overview 61
- Terminology Q&A 63
- Design Steps 64
- Accessing Client Device Functionality 66
- XPath in MobileTogether 67
- RecordsManager 70

## File paths in Windows

File paths given in this documentation will not be the same for all operating systems. You should note the following correspondences:

- (My) Documents folder: Located by default at the following locations. Example files are located in a sub-folder of this folder.

| | |
|---|---|
| Windows 7/8/10/11 | `C:\Users\<username>\Documents` |

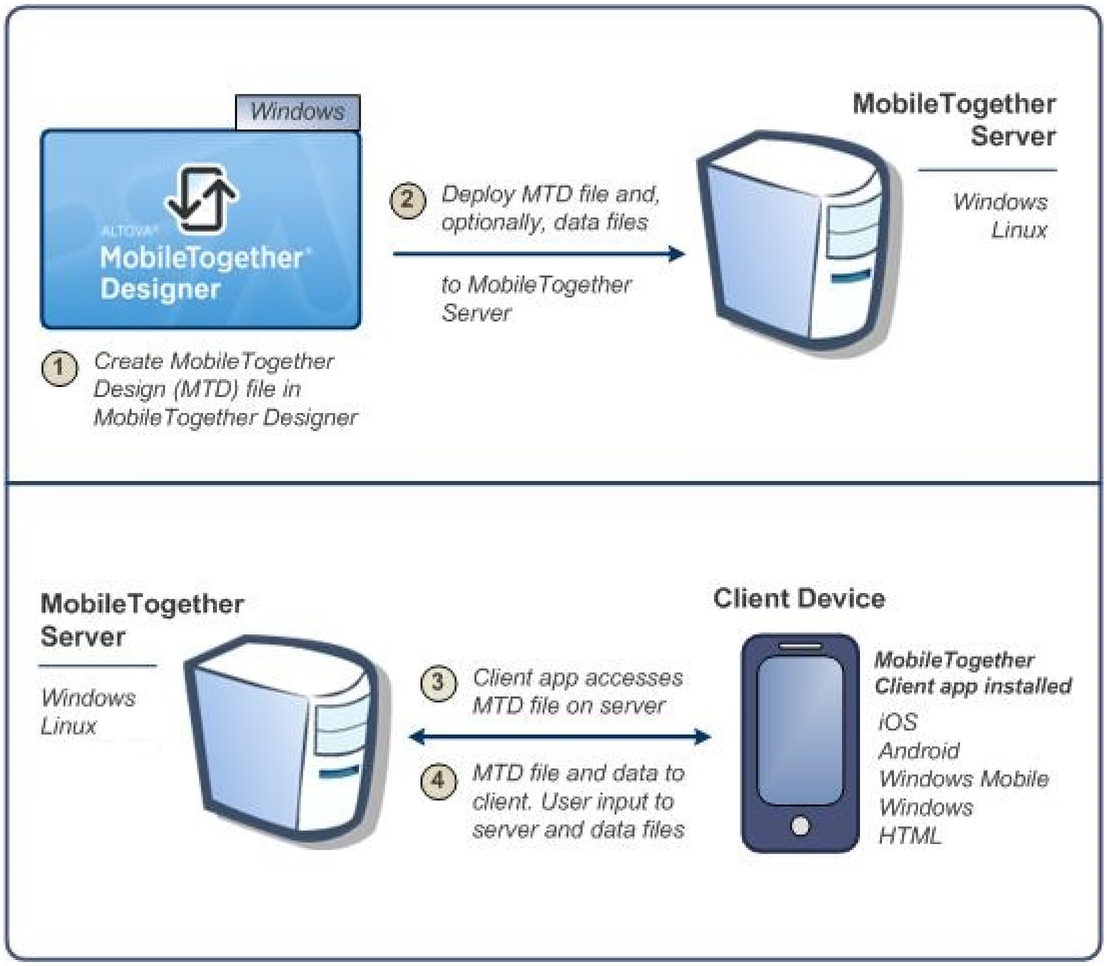
- *Application folder:* The Application folder is the folder where your Altova application is located. The path to the Application folder is, by default, the following.

| | |
|---|---|
| Windows 7/8/10/11 | `C:\Program Files\Altova\` |
| 32-bit version on 64-bit OS | `C:\Program Files (x86)\Altova\` |

## 3.1 MobileTogether Overview

MobileTogether consists of the following modules:

- *MobileTogether Designer*, in which MobileTogether solutions for mobile clients (MTD files with the extension `.mtd`) are created. These MobileTogether solutions are then uploaded to MobileTogether Server.
- *MobileTogether Server*, which serves MobileTogether solutions to mobile clients.
- *MobileTogether Client* apps (for iOS, Android, Windows Phone 8, Windows RT, Windows Metro, web clients, web-based smartphones/tablets), on which the end user receives and interacts with MobileTogether solutions (**`.mtd`** files) delivered by MobileTogether Server.


Windows
ALTOVA
MobileTogether Designer
1 Create MobileTogether Design (MTD) file in MobileTogether Designer
2 Deploy MTD file and, optionally, data files
to MobileTogether Server
MobileTogether Server
Windows
Linux
MobileTogether Server
Windows
Linux
3 Client app accesses MTD file on server
4 MTD file and data to client. User input to server and data files
Client Device
MobileTogether Client app installed
iOS
Android
Windows Mobile
Windows
HTML

### System requirements

▼ MobileTogether Designer

| | |
|---|---|
| Windows | Windows 10, Windows 11 |

| | |
|---|---|
| Windows Server | Windows Server 2016 or newer |

### MobileTogether Server

| | |
|---|---|
| Windows | Windows 10, Windows 11 |
| Windows Server | Windows Server 2016 or newer |
| Linux | • Red Hat Enterprise Linux 7 or newer<br>• CentOS 7, CentOS Stream 8<br>• Debian 10 or newer<br>• Ubuntu 20.04, 22.04, 24.04<br>• AlmaLinux 9.0<br>• Rocky Linux 9.0 |

### MobileTogether Client

| | |
|---|---|
| iOS | 15 and higher for Apple mobile devices |
| Android | 5.0 and higher for Android mobile devices |
| Windows RT, Metro | Windows 10; Windows RT for Windows touch-enabled PCs and tablet computers |
| HTML | HTML browsers for any other mobile devices |

## 3.2 Terminology Q&A

### How does the MobileTogether system work?

- In **MobileTogether Designer**, you create **MobileTogether Design files (MTD files)**, which have the file extension **`.mtd`**.
- These files are deployed to a **MobileTogether Server**, from where they are served to the **mobile client device** as **MobileTogether solutions**.
- The data files that are used to populate the design template/s in the MTD file may reside at their original locations or can be deployed to MobileTogether Server together with the MTD file.
- On the mobile client, the **end user** can view reports presented in a layout that is defined in the MTD file. End users can also use the MobileTogether solutions on their mobile client devices to update data files at their server locations.

### What's in an MTD file and in a MobileTogether project?

- An MTD file is a native MobileTogether Designer document.
- Each MTD file contains one **MobileTogether project**.
- A MobileTogether project consists of one or more **pages**[380]. A page is what the end user sees on the mobile client device.
- If there is more than one page in the MTD file, then these pages are connected to one another in a simple sequence, with the first page leading to the next, and so on, till the last page is reached.
- Sub-pages[257] can also be defined, and these can be accessed from within main pages with the `GoToSubpage` action.

### What does a page consist of?

- A page consists of **page controls**[403] (also called **'controls'** for short), formatted for viewing on the mobile client device and set up for user-interaction.
- Each control has different properties. These properties define associated content, formatting, and **action/s**[667] to perform when an event of a control (**control event**[665], for short) is triggered.
- For each page, a set of page sources can be defined in the Page Sources Pane[270] of that page.
- The content associated with a control can come from one (or more) of these **page sources**. Such data is accessed using the XPath/XQuery language.
- Via its controls, therefore, a page presents data to the end user and can accept modifications to its data sources.

### What are the different kinds of events and actions in a design?

- *Control events and their actions*[665]*:* Each control on a page can have events that trigger actions you can specify. For example, the combo box control has the `OnFinishEditing` event, which occurs when an item from the dropdown list of the combo box is selected. This event can be defined by you to trigger a desired action, such as changing data as a result of the combo box selection.
- *Page events and their actions*[389]*:* The page itself (as a single entity) can be associated with events that trigger actions. For example, `OnPageLoad` is a page event. This event can be defined by you to trigger a desired action, such as loading data into the page from a certain data file.

## 3.3 Design Steps

Given below is a broad outline, in steps, of how to create a MobileTogether Design file (MTD file).

1. Create a new MTD file

   Each MTD file represents a project consisting of one or more pages in a simple sequence. When a new MTD file is created, it has one default page that has no page source. You can add page sources (data sources) to the default page, and you can add more pages to the project (*see the points below*). Create a new MTD file with the **File | New** 1562 command. The file is created in memory and must be saved with the **File | Save** 1568 command to store it on disk. Define Project Properties 296 and specify other project-related settings 286.

2. Add data sources for the page (page sources)

   Each page is assigned data sources, from which it obtains the data that will be displayed in the page. The data sources of a page are added via the Page Sources Pane 270 as page sources, and each page source is shown there as a tree of nodes. Data from these nodes is used by the controls in the page design, for display, or for processing that leads to some kind of data representation (such as charts or images). Nodes in page source trees are addressed using XPath expressions. Client data input can also be saved back to the page sources if desired. See the section Page Sources 315 for details.

3. Add controls to the page, and define their properties and event-actions

   Page controls are added to a page from the Controls Pane 266. Each control has a set of properties (defined in the Styles & Properties Pane) 274 and data (from the page source trees 315) associated with it. A control can also have one or more predefined events. You can specify the action/s to be performed when a control event is triggered. For example, a button control has the event `OnButtonClicked`, and this event can have an `Open URL` action associated with it. For more information, see the sections Page Events 389 and Actions 667. Additionally, pages also have events, and you can specify actions to perform when a page event is triggered. For example, when a page is loaded (a page event), an action can be specified that loads data from a specified XML file into a given page source.

4. If required, add more pages to the project and design these

   Additional pages can be added to the initial page. A new page can be added as a top page or a sub page by clicking the **Add Page** icon in the Page Pane's toolbar 257. The sequence of top pages in the Pages Pane 257 determines the sequence of the workflow.

5. Create a flow between top pages and sub pages

   You can further structure the solution's workflow by using sub pages. These are accessed from within top pages with the `GoToSubpage` 667 action (of control or page events). Other page-related actions 667 provide for more movement between pages.

6. Optionally, add additional design and user-related functionality to the project

   After all the pages have been added and the structure of the workflow has been finalized, you can revise your page designs and workflow. Any additional design components or actions can be inserted in the project now.

7. Run a simulation of the MobileTogether solution

   You can test the design by running a workflow simulation[1355] within MobileTogether Designer. The simulation shows (in MobileTogether Designer itself) how the workflow will be executed on the client device. Select **Run | Simulate Workflow** or press **F5** to start the simulation. The Messages Pane[1385] provides a detailed and step-by-step report of workflow activity, enabling effective and easy debugging.

8. Deploy the MTD file to MobileTogether Server

   After making final changes and re-testing the file, save it, and then deploy it[291] to MobileTogether Server. The MobileTogether solution is now ready to be accessed by mobile client devices.

9. Optionally, create the solution as an AppStore App

   You can create a MobileTogether custom app that end users can download to mobile devices. We call these apps AppStore Apps. The section AppStore Apps[1471] describes how to generate the program code for such apps from your MobileTogether Designer project. Code can be generated for Android, iOS, Windows (touch-enabled devices and PCs), and Windows Phone. After the code has been generated, it can be compiled into the corresponding AppStore App.

## 3.4 Accessing Client Functionality

MobileTogether solutions use or leverage the functionality of client devices to provide their unique benefits. Access to client functionality is provided by actions 667 that you add to the design. The following client device features are the most important of those that may be accessed by a MobileTogether solution:

- *WiFi and LAN:* A solution would typically use the device's connectivity features.
- *Camera:* The device's camera can be used for actions as diverse as recording video to scanning a barcode.
- *Gallery:* Images can be saved to or loaded from the device's photo gallery.
- *Microphone:* The device's microphone can be used for audio recordings.
- *NFC:* The device's NFC capability can be used by the NFC actions 744.
- *GPS:* The device's location services can be used in Geolocation Services actions 733.
- *Contacts:* The device's contacts can be accessed.
- *Calendar:* The device's calendar can be accessed.
- *Telephony and SMS:* The device's telephony and SMS functionality can be accessed.
- *Files:* Files stored on the device can be accessed.

## 3.5 XPath in MobileTogether

The XPath language plays a crucial part in the design of MobileTogether solutions. XPath is used to locate, access, manipulate, generate, and save data in the various data trees used in the design and to define the functioning of different design components. Some important ways in which XPath is used in a MobileTogether design are given below. This overview is intended to give you a broad sense of the flexibility and power of XPath and of how XPath is used in MobileTogether designs.

For more information about XPath, see the XPath 3.1 Recommendation of the W3C, which is the latest version of the language available and is the version supported in MobileTogether Designer. To get you started using a more learning-based approach, see the following:

- Altova's A Gentle Introduction to XPath
- Altova's XPath 3.0 Training
- W3C's XPath Tutorial

### Locator expressions

The locator expressions of the XPath language are used to locate nodes in XML trees. A locator expression typically consists of a path that locates the required node. Here are some examples:

- **`/Company/Office`**: Locates all `Office` child elements of the `Company` element, which is the top-level document node. We know that the Company element is the top-level element because it occurs directly under the root node, which is indicated by the first forward slash.
- **`/Company/Office[3]`**: Locates the third `Office` child element of the `Company` element.
- **`/Company/Office[3]/@location`**: Locates the `location` attribute of the third `Office` child element of the `Company` element.
- **`//Office[@location='US']`**: Locates all `Office` elements that have a `location` attribute having a value of `US`.

The list above shows just a few basic locator expressions. There are many more ways in which locator expressions can be constructed.

### Operators

Operators allow you to apply filters, build conditions, and manipulate selections and data. For example, here are just two operators:

- **`if (Selection='US') then //Office[@location='US'] else //Office[@location!='US']`**: This `if` operator selects US or non-US offices depending on the content of the `Selection` child element.
- **`for $i in //Office return $i[@location='US']`**: This `for` operator returns all `Office` elements that have a `location` attribute having a value of `US`.

### XPath functions

XPath functions enable the manipulation, calculation, and generation of data. For example, a function can take a string as input (the function's argument), and convert into lowercase or even remove a part of the string. The XPath functions that can be used in MobileTogether designs are of the following types:

⊟ *Built-in functions*

The XPath language contains a large library of built-in functions which enable you to extract data as well as metadata related to the XML tree, and even to generate data. For example:

- **`count(Office)`**: Returns the number of `Office` child elements.
- **`day-from-date("2015-04-26")`**: Returns the number `26`, which is the day part of the function's date argument.

User guides and references for the built-in functions are widely available on the Internet. A full list of the functions can be found in the XPath 3.1 Recommendation of the W3C.

⊟ *Altova extension functions*

This is a set of of XPath extension functions that Altova is developing to provide developers with more functionality in XPath. There are currently some 60 extension functions 1689, ranging from functions that provide geolocation information to those that convert integers to hexadecimal strings and vice-versa. For example:

- **`format-geolocation(33.33, -22.22, 2)`** returns the xs:string `"33.33N 22.22W"`
- **`hex-string-to-integer('1')`** returns `1`

Altova extension functions are available for use in all MobileTogether designs. For usage information, see the Altova Extension Functions 1689 section in the MobileTogether Designer user manual.

⊟ *MobileTogether extension functions*

These are XPath extension functions developed by Altova for specific uses in MobileTogether designs. For example:

- **`mt-has-server-access(10)`** returns `true` if server access is possible within the time in seconds that is specified as the argument of the function, `false` otherwise.
- **`mt-load-string('MyCourier')`** returns the localized `MyCourier` string that is stored in the solution's string pool. The language of the localization is selected automatically according to the language of the mobile device.

MobileTogether extension functions are available for use in all MobileTogether designs. For usage information, see the MobileTogether Extension Functions 1262 section in the MobileTogether Designer user manual.

⊟ *User-defined extension functions*

These are XPath extension functions that you, the user, can define in a design for some special purpose you have in mind and for which no suitable function exists in the function libraries listed above. For example, you might want to define a function to convert temperatures between the Celsius and Fahrenheit scales. User-defined functions are defined within a single MobileTogether project and are used in that specific project. How to define such custom functions is described in the User-Defined XPath/XQuery Functions 1293 section of the MobileTogether Designer user manual.

## Global variables

Global variables contain information about the client mobile device. For example, there is one variable to indicate the device's type, another to indicate its dimensions, and yet another to indicate the device's current orientation (landscape or portrait), and so on. The values of all these variables are obtained at run-time from the client device as part of standard mobile communication procedures. Variables can then be used in XPath/XQuery expressions. As a result, processing can be specified that is conditional upon a device's inherent static properties (such as size) or its changeable dynamic properties (such as orientation).

MobileTogether's global variables are predefined and are listed in the section Global Variables 1298 together with each variable's description and possible values. The example below of the `MT_iPad` global variable (possible values: `true()`, `false()`) shows how global variables are called in XPath expressions. The `$` symbol is used to indicate that what follows is the name of a global variable, which is the usual way to indicate variables in XPath.

```
if ( $MT_iPad=true() ) then "Apple" else ""
```

## 3.6 Altova RecordsManager

Altova RecordsManager™ (in short, also RecordsManager) enables you to build business database solutions in record time using a powerful visual design interface. You can read more about RecordsManager at the Altova website.

RecordsManager can be deployed as a solution to a MobileTogether Server and/or generated as an AppStore App[1471]. You can access RecordsManager in either of these formats to design a database and make it available for use among multiple users. RecordsManager provides complete management of records (such as contracts), with features such as templates for documenting record details, built-in reminders, change logs, and change tracking. A centralized repository that is accessed via the Internet provides numerous benefits, from time savings to efficient management of related records and automatic reminders of important record-related dates.

RecordsManager is available in your installation of MobileTogether Designer (version 8.0 onwards) as a MobileTogether Package[295]. You can use the RecordsManager package as follows:

- Run a simulation in MobileTogether Designer to try out RecordsManager.
- Deploy RecordsManager to your MobileTogether Server.
- Create an AppStore App[1471] from the RecordsManager package.

### Admin credentials

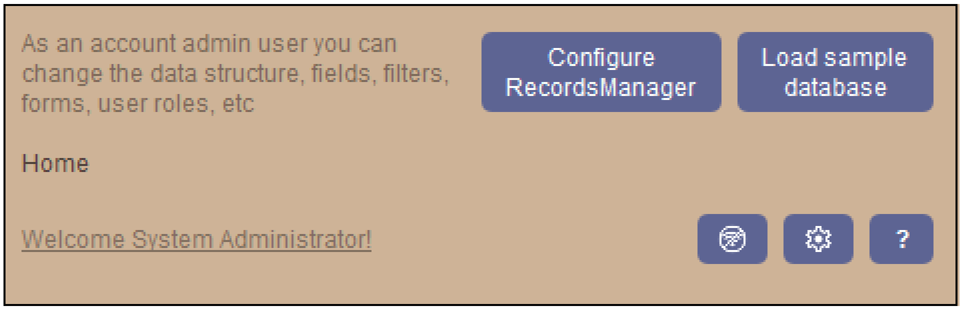
When you start RecordsManager, whether as a simulation or as a solution, you can do so with administrator credentials. The initial credentials are:

*Login:* **`root`**
*Password:* **`root`**

After you log in as an administrator, you can set up new users with varying privileges. For information, see the Altova RecordsManager Admin Guide.

### Simulate RecordsManager

To start a RecordsManager simulation in MobileTogether Designer, select the menu command **Run | Run RecordsManager**[1625] (or click this command's icon in the main toolbar). In the RecordsManager login screen that appears, enter the credentials given above (**`root/root`**). On successful login, the RecordsManager Home Page appears (*top part shown in screenshot below*).



As an account admin user you can change the data structure, fields, filters, forms, user roles, etc

Home

Welcome System Administrator!

There are a number of pre-built sample databases that you can use to get an idea of how RecordsManager works. To select a sample database, on the RecordsManager Home Page, click **Load sample database** (*see screenshot above*).

For information about simulations, see the description here[1355].

## Deploy to MT Server

After you have opened the RecordsManager package in MobileTogether Designer (*see above*), make sure that the simulation has been closed. With the **`RecordsManager.mtp`** file active, select the menu command **File | Deploy to MobileTogether Server**[1574] and deploy as you normally deploy a solution[291]. After RecordsManager has been deployed, you can access it just like any other solution. Log in as an admin using the credentials given above (**`root/root`**) and work with RecordsManager just as you did in simulations.

## Create as AppStore App

In MobileTogether Designer, you can also create an AppStore App[1471] from the RecordsManager package (**`RecordsManager.mtp`**). Make sure that the simulation has been closed and then select the menu command **File | Generate Program Code for AppStore Apps**[1580] to do this. Create an AppStore App[1471] as usual (described here[1472]). When you download and start the RecordsManager App Store App, you will get the login screen familiar from your simulations (*see above*). Work with RecordsManager as usual.

# 4 Tutorials

This section contains tutorials that will guide you through the basic steps needed to create a MobileTogether design, as well as help you to understand more advanced MobileTogether features.

- The QuickStart Tutorials 74 take you through the basics, all the way up to deploying the MTD file and associated files on MobileTogether Server.
- The Simple Database tutorial 107 shows how to create a DB-based design and how to load DB records on the basis of a user selection.
- The Hierarchical Database 118 explains how related tables of a DB can be hierarchically linked in a page source.
- The Database-And-Charts tutorial 159 uses a finished design file to explain how to work with databases and how to create charts.
- The SubPages-And-Visibility 188 tutorial shows you how to open a sub page from a top page, and how to filter the display of a data structure using the `Visible` property.
- The Add and Edit Records 206 tutorial explains a number of intermediate-level concepts that will help you gain more confidence in using MobileTogether Designer.
- The SOAP Requests 215 tutorial explains how to create SOAP requests to obtain data from a web service, and how to use the returned data in the design.
- The Sharing Geolocations 230 tutorial shows how the Geolocation functionality can be used. Other features that are described include sharing 699 and catching errors 907.
- The Scrollable Tables 236 tutorial describes the key features of scrollable tables, and shows how such tables are created.

## Tutorial and example files

Finished design files, as well as the data sources and image files used in the tutorials, are available in the *(My) Documents* folder: `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials`.

We recommend that you create the folder **`C:\MobileTogether`** and then copy all the folders and files that are in the Tutorials folder to the **`C:\MobileTogether`** folder. This is because some of the supplied design files (**`.mtd`** files) use absolute URLs to target resources in the **`C:\MobileTogether`** folder. If, while deploying a design file to the server, some resource file that must be deployed (such as an image file) cannot be correctly located, then (i) remove that resource file from the list of files to deploy, (ii) browse for the resource at the location where it is saved, and (iii) add the resource file (now with the correct location) to the list of files to deploy.

After you have completed the tutorials, you could open the more advanced example design files (**`.mtd`** files) in the `MobileTogetherDesignerExamples` folder. These examples show how various features of MobileTogether Designer can be used, and so can be used as reference points for your own designs.

## Video demos of how to get started with MobileTogether

The Altova website provides video demos that will help you to get started with MobileTogether Designer and to understand some of its key features. Currently, the following video demos are available:

- *Intro and building your first app*: Describes the user interface and the the first steps to take when creating a design
- *Building a database-driven app*: How to connect to a DB and query it, how to retrieve DB records, and how to present DB data in the form of tables

- *Working with databases, Part 2*: Adds functionality to the design created in the previous demo: how to query, view, and edit DB records
- *Working with databases, Part 3*: Lets users upload images, resize images, and save the edited images; builds on the previous two demos
- *Working with databases, Part 4*: Adds functionality (to the previous demo) for record creation and deletion, and for data validation
- *Working with databases, Part 5*: Adds (to the previous demo) filtering functionality without the need for calls to a backend server; plus how to create user-defined XQuery functions
- *Install and Configure MobileTogether Server*: Shows how to install MobileTogether Server and Altova LicenseServer, and how to configure MobileTogether Server behind a corporate firewall
- *Configuring MobileTogether Server in a Network*: Also explains how to set up ports so that MobileTogether Sever can be connected to from both outside and inside the network
- *Configuring MobileTogether Server to use SSL*: Describes how to enable secure connections between MobileTogether Sever and client devices. Includes descriptions for: (i) purchased certificates and (ii) Let's Encrypt certificates
- *An Altova blogpost* about configuring MobileTogether Server in a network
- *Plus*: Additional demos about using charts and working with XQuery

## File paths in Windows

File paths given in this documentation will not be the same for all operating systems. You should note the following correspondences:

- (My) Documents folder: Located by default at the following locations. Example files are located in a sub-folder of this folder.

| | |
|---|---|
| Windows 7/8/10/11 | `C:\Users\<username>\Documents` |

- *Application folder:* The Application folder is the folder where your Altova application is located. The path to the Application folder is, by default, the following.

| | |
|---|---|
| Windows 7/8/10/11 | `C:\Program Files\Altova\` |
| 32-bit version on 64-bit OS | `C:\Program Files (x86)\Altova\` |

**File URLs in example files**
File URLs in example files might be absolute URLs. In such cases, they will probably not correspond to the directory structures of your work environment. If you use the supplied example files, make sure that file URLs in them—and any XPath expressions that build such URLs—point to the correct file locations on your machine or network.

# 4.1 QuickStart (Part 1)

This QuickStart tutorial guides you through the basic steps of creating a MobileTogether design for a MobileTogether solution. The solution displays the splash screen of an Altova product which the end user selects in a combo box. The tutorial shows you how to set up a page, its page sources, how to add controls, and make the display of the splash screen conditional upon end-user input. It also explains validation, how to deploy the design file to MobileTogether Server, and how to run simulations. After you have finished with this QuickStart tutorial, you should have a good understanding of the broad working principles of MobileTogether designs. You will then be ready to tackle more complex designs.

This tutorial is organized into the following sections, each of which deals with one key aspect of creating a MobileTogether design.

- Create a New Design 74
- Set Up a Page 76
- Add a Page Source (or Data Source) 77
- Format the Design 80
- Add a Control: Combo Box 81
- Add a Control: Image 83
- Define Control Actions 86
- Validate the Project 89
- Run a Simulation 89
- Deploy to Server 92

## The tutorial files

The files for this tutorial are located in your ( 72 *My) Documents* 72 MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\QuickStart**. You can save the splash screen image files that are used in the tutorial to any other location and use them from there if you like.

- The file you will end with should be similar to: `QuickStart01.mtd`
- The image files used in the tutorial are the `*.bmp` files in the QuickStart tutorial folder

## Video demo of how to build your first app

The Altova website provides a video demo that describes the user interface and explains the first steps to take when creating a design.

## 4.1.1 Create a New Design

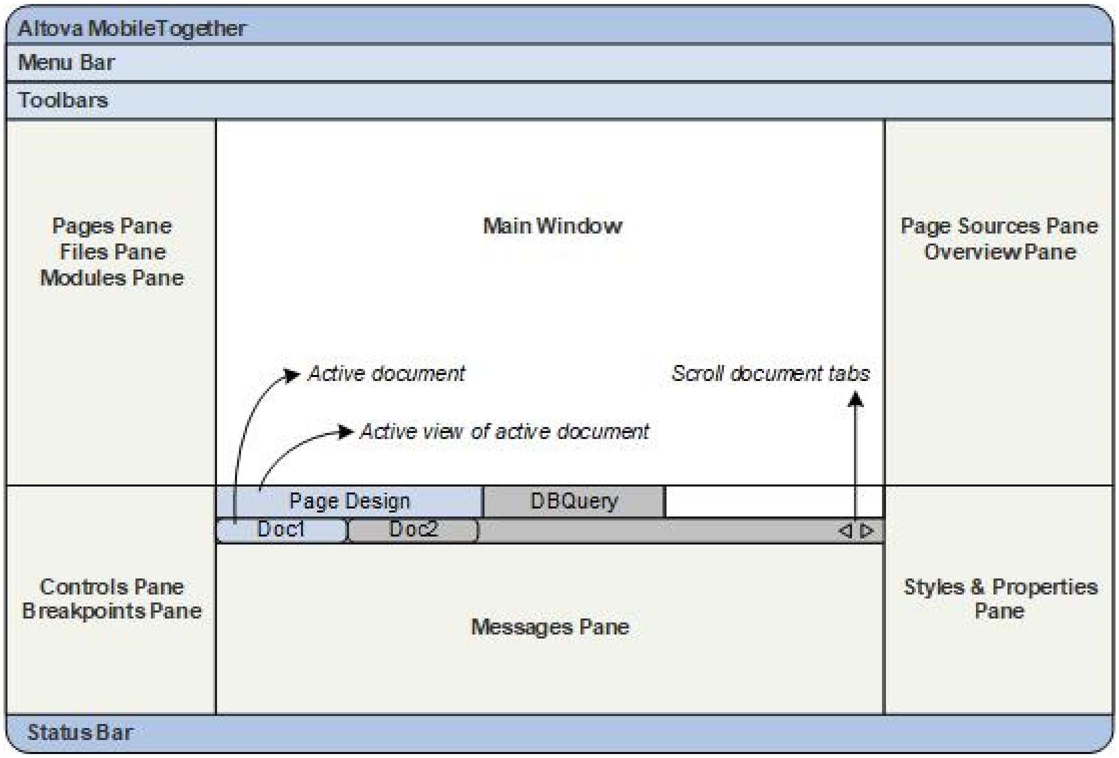
*In this part, you will:*

- Create a new MobileTogether design
- Save the design to file
- Set up the design's preview properties: the preview device and magnification level
- Familiarize yourself with the GUI of MobileTogether Designer

A MobileTogether design is created in an MTD (MobileTogether Design) file, which has the **`.mtd`** extension and is native to MobileTogether Designer.

To create a new MobileTogether design, do the following:

1. Click **File | New**. This opens a new design file in the main window of the GUI (*figure below*). It will have a default page called `New Page1`.



2. The new design will be in a tab called `New Design1`. Click **File | Save**, and save the file with any name to any location you like. Make sure, however, that you save your design file in the same folder as the splash screen images. (Since your design, when complete, will show the splash screens of Altova products, you could, for example, call it `AltovaSplashScreens.mtd`.) The new file name, which will have a `.mtd` extension, replaces `New Design1` in the file's tab. The file name also appears in the application's title bar.
3. In the Main toolbar of MobileTogether Designer 254, you can select your **preview device** and set the **magnification** of the design. Select the options you want. Since mobile devices have different background colors and dimensions, selecting an appropriate preview device will help you visualize your design better. You can change the preview device at any time if you wish to visualize the design on another device. Note that these settings will also be applied to simulations 89.
4. Familiarize yourself with the Main Window and its tabs 253, and with the different panes that are located around the main window 251. The user interface is described in the section, The User Interface 251.
5. Switch between the Page Design 253 and DB Query 255 tabs of the Main Window 253, and see how the contents of the panes change.

## 4.1.2 Set Up a Page

A MobileTogether design can consist of one or more pages. A page is what the end-user sees on the mobile client device. If there are multiple pages in a design, the page sequence is defined by their sequential order in the Pages Pane 257. Within a page, controls can be set to go to sub pages (that typically contain reusable modules). In this part of the tutorial, in order to keep things simple, we will create a project that has only one page. For more information about pages, see the description of the Pages Pane 257.

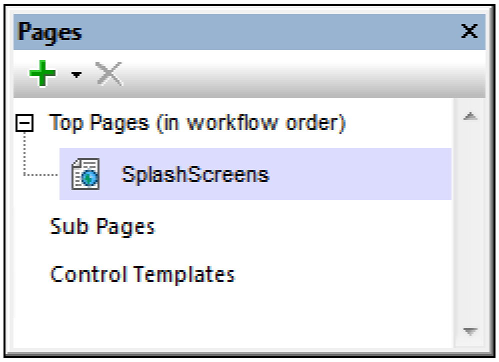
*We are building a design that has one page. In this part, you will:*

- Give the default page a name
- Add a label control so as to display a title for the page
- Format this label

### Page name and the label control

Give the default page of the new design a name as follows:

1. In the Pages Pane 257 (*screenshot below*), double-click `New Page1` and rename it to `SplashScreens`.



2. From the Controls Pane 266, drag and drop the Label control 554 into the design. The control will be placed at the top of the page. A red exclamation mark appears in the control when you click anywhere outside the control. On hovering over the exclamation mark, a message is displayed, warning that the label has no content.
3. The content of the label can be static or dynamic. If dynamic, the content can be taken from an XPath expression or from a node in one of the page's page sources. (No page source has been defined for the `SplashScreens` page as yet. This will be done in the next part 77.) Enter static content by double-clicking the label and entering the text, `Altova Splash Screens`, and then pressing **Enter**.
4. In the Table toolbar, click **Center** to center the text. Then click **Text Color** and/or **Background Color** (also in the Table toolbar) if you wish to set these properties.
5. With the Label control 554 selected, the label's properties are displayed in the Styles & Properties Pane 274. If you like, modify any of the label's formatting properties 554 available in this pane, such as the margin-bottom property to create empty space below the label.

### 4.1.3 Add a Page Source (or Data Source)

*You have so far created a page and given it a title 76. In this part, you will:*

- Add an empty XML page source for this page
- Since this page source has neither structure or content, create structure and content directly in the Page Sources Pane 270
- Set a default XPath context node

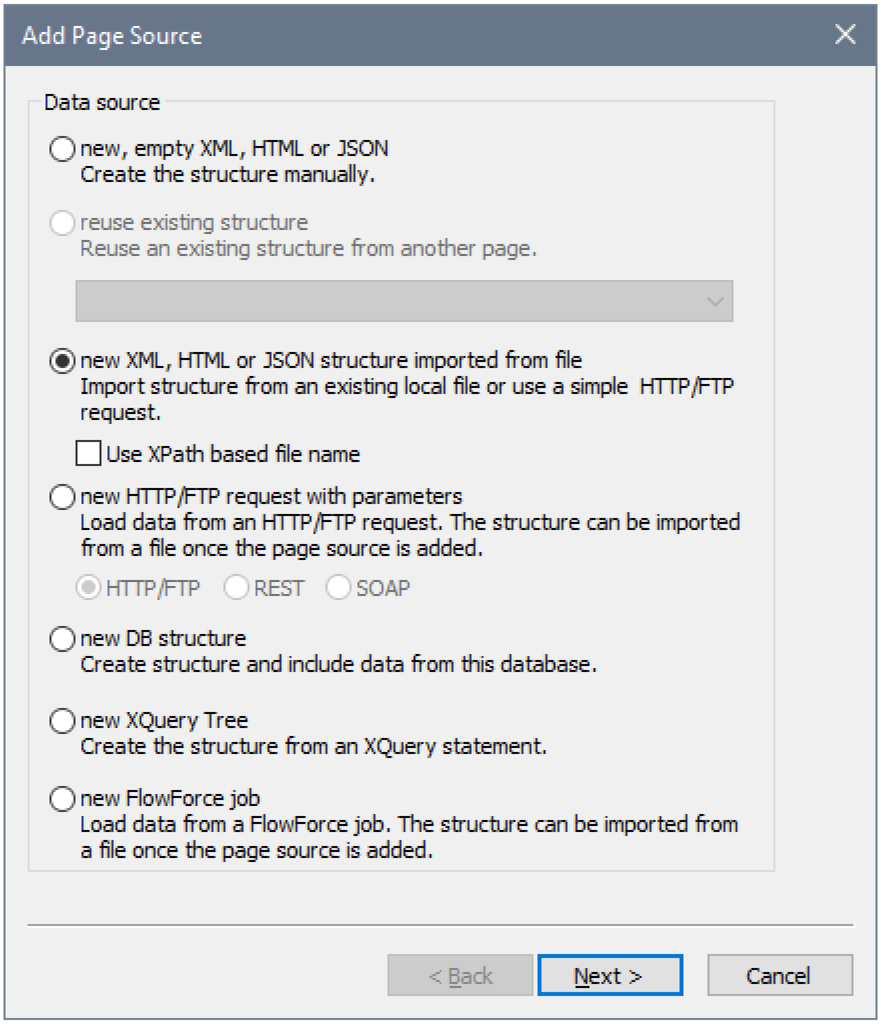
⊟ *Buttons in this section*

➕ **Add Source**

A page can have one or more page sources from where data can be obtained. Page sources are defined in the Page Sources Pane 270 and can be specified to be read-only or editable. They can be external sources, or can be created directly in the Page Sources Pane 270 and contain static data. The types of page sources that can be used are described in the section Page Sources (Data Sources) 315. In this tutorial, we will add one type of page source, an editable empty XML source, for which we will then create a structure and content directly in the Page Sources Pane 270.

To add the page source for the page, do the following:

1. Click the **Add Source** button in the toolbar of the Page Sources Pane 270 to display the Add Page Source dialog (*screenshot below*).



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

● new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

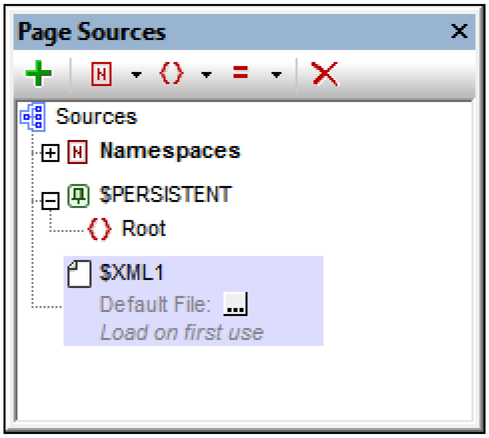
● HTTP/FTP ○ REST ○ SOAP

○ new DB structure
Create structure and include data from this database.

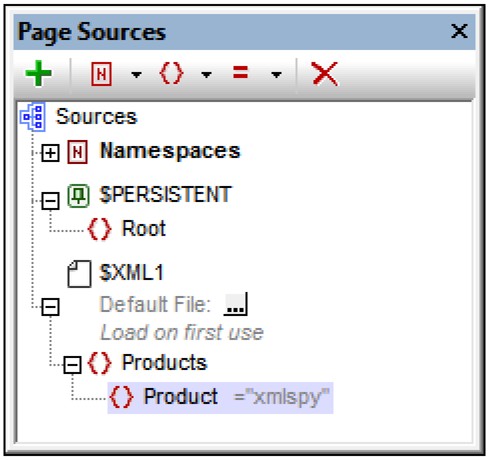
○ new XQuery Tree
Create the structure from an XQuery statement.

○ new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

2. Select *New, empty XML*, and click **Next**.
3. In the next screen, keep the default settings (*XML, Editable, Data is copied to client, Load data on first use*), and click **OK**. (When a page source is created as editable, data in it can be modified.) A root node called `$XML1` 349 is added to the Page Sources tree 319 in the Page Sources Pane 270 (*screenshot below*).



4. Notice that the XML tree with the root node `$XML1` is empty. Right-click `$XML1` and select **Add Child | Element,** then enter the element name, `Products`.
5. Add a child element to `Products` (by right-clicking it and selecting **Add Child | Element**), and name the child element `Product`. If you wish to rename the elements, double-click the element names and edit them.
6. Right-click the `Product` element, select **Ensure exists on load (fixed value)**, then enter `xmlspy` (*see screenshot below*). This will be the `Product` element's default content when the page loads.



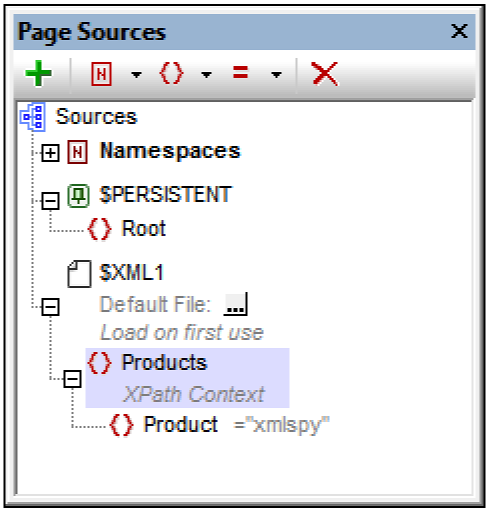
Notice that there is no default XML file assigned to `$XML1`. A default file would provide the data that goes into the nodes of the page source's tree. The data provided by a default file can be overridden by data that is manually entered in a tree node. In our case, there is a single data node: `$XML1/Products/Product`, and it has the string `xmlspy` as its default content. The XML data structure is:

```
<Products>
   <Product>xmlspy</Product>
</Products>
```

## Set a Page XPath Context Node

Each page has a Page XPath Context Node, which is used as the context node for relative XPath expressions defined on that page. The Page XPath Context Node is indicated in the Page Sources Pane [270] with the label *XPath Context* (*see screenshot below*). Note that each page has its own Page XPath Context Node.

We will change the Page XPath Context Node to the **`Products`** node. Do this by right-clicking **`Products`** and selecting **Set As Page XPath Context**. The element **`Products`** will be set as the Page XPath Context Node—this is indicated by the node being labeled *XPath Context* (*see screenshot below*).



**Note:** If you wish, in your XPath expression, to reference a node in a page source tree other than that of the Page XPath Context Node, use an absolute path to that node, starting with the root node of that tree 349. For example: **`$XML2/SomeRootElement/SomeOtherElement`**.

## 4.1.4 Format the Design

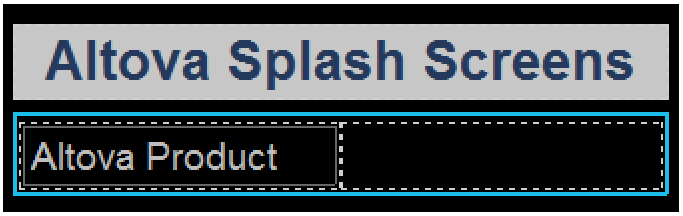
*The design page now has a name (SplashScreens 76), a display title (Altova Splash Screens 76), and a page source ($XML1 77). In this part, you will:*

- Add a Table control for presentation purposes
- Add formatting: the horizontal line control, spacing, and color
- Copy-paste controls to other parts of the design

### Add a table

We will add a table for presentation purposes.

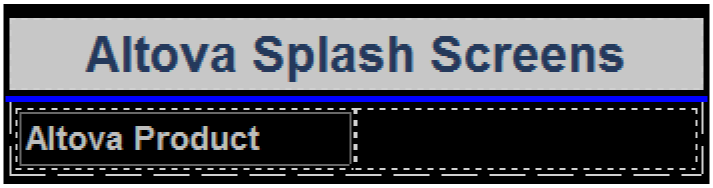
1. From the Controls Pane 266, drag the Table control 614 into the design and drop it below the label.
2. In the New Table dialog that appears, specify that the table should have two static columns and one static row, and then click **OK**. The table is created with a single row and two columns.
3. From the Controls Pane 266, drag and drop the Label control 554 into the left-hand cell, and give it a static text value of **`Altova Product`** (*see screenshot below, which shows the table highlighted*).



### Add a horizontal line, spacing, and color

Add a horizontal line to separate the title from the table as follows:

1. From the Controls Pane[266], drag and drop the Horizontal Line control[459] between the title and the table (*see screenshot below; the horizontal line is dark blue*).



2. With the line selected in the design, in the Styles & Properties Pane[274], set the color and width of the line.
3. Copy the line, by selecting it and pressing **Ctrl+C**, and paste it below the table (using **Ctrl+V**). The line will be copied with all the properties that you defined for it.
4. You can change the background color of the label, the table, and individual table cells. Try this by placing the cursor in the respective components, and selecting different background colors in the Styles & Properties Pane[274].
5. Select the label control and set `Margin Top` and `Margin Bottom` properties in the Styles & Properties Pane[274].
6. Try copying different components to various parts of the design.

## 4.1.5 Add a Control: Combo Box

We will add a combo box to the right-hand cell of the table you created in the previous section. We will then define the properties of the combo box. A combo box needs two important property definitions:

- *The combo box values:* These values will populate the dropdown list of the combo box and provide the options the user can choose (*visible entries*). Each visible entry will have a corresponding XML value that will go into the associated XML node.
- *The associated XML node:* This page source node is synced with the combo box selection. It receives its value from the combo box selection. Its initial value determines the initial combo box selection.

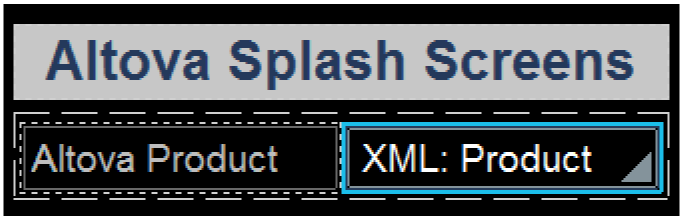
⊟ *Buttons in this section*

| | |
|---|---|
| ... | **Additional Dialog** |

## Add a combo box

Add the combo box and define its properties as follows:

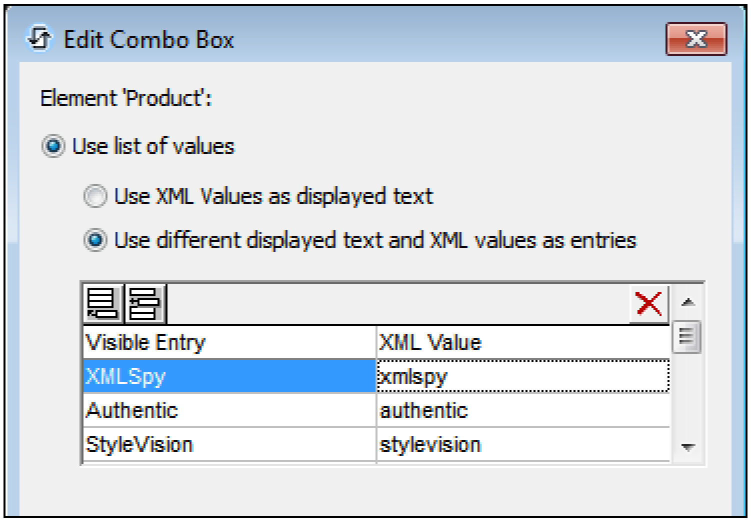
1. From the Controls Pane[266], drag and drop the Combo Box control[459] into the right-hand cell of the table (*see screenshot below*).



Altova Splash Screens

Altova Product | XML: Product

2. Drag the `Product` node from the Page Sources Pane[270], and drop it on to the combo box. This associates the `Product` node with the combo box, and creates what we call a source node (or page source link). When the end user selects an option from the combo box, the XML value of the selected option will be passed to the `Product` node and will become the `Product` node's content.
3. To define the dropdown list items of the combo box, select the combo box, and, in the Styles & Properties Pane[274], click the **Additional Dialog** button of the *Combo Box Entry Values* property. The Edit Combo Box dialog (screenshot below) appears.



Edit Combo Box

Element 'Product':

● Use list of values

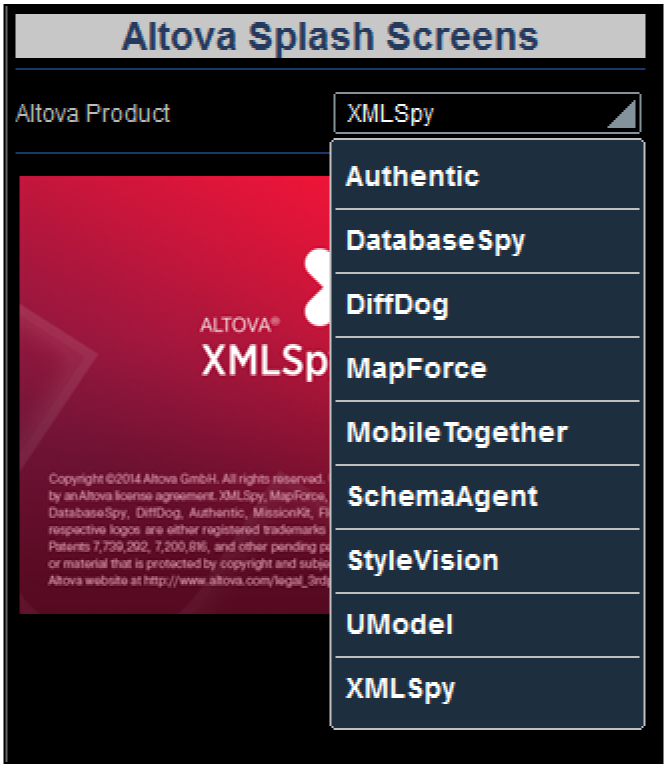
○ Use XML Values as displayed text

● Use different displayed text and XML values as entries

| Visible Entry | XML Value |
|---|---|
| XMLSpy | xmlspy |
| Authentic | authentic |
| StyleVision | stylevision |

4. Select *Use list of values*, and then *Use different text and XML values*. Enter values for the visible entry (which is what will appear in the dropdown list of the combo box) and for the corresponding XML value (which is what will be written into the `Product` node). When we add the Image control[83], you will find out why we want the XML values lowercased. (Hint: The image names are lowercased, for example, `xmlspy.bmp`.) You can add up to nine Altova products to this list in any order you like: XMLSpy, Authentic, StyleVision, MapForce, DiffDog, DatabaseSpy, MobileTogether, SchemaAgent, UModel.
5. Select the *Sort* values check box at the bottom of the dialog to sort the list when it is displayed, and click **OK** to finish.

When the completed solution is executed or a simulation of the completed project is run[1355], the dropdown list of the combo box (*screenshot below*) will display the values listed in the *Visible Entry* column definitions (*see screenshot above*).



The objective of our project is this: When the end user selects an entry from the dropdown list, the XML value corresponding to that entry (*see screenshot of the combo box definition above*) will be passed to the `Product` node of the page source. Note that the default content of `Product` is `xmlspy` (defined when the node was created[76]). We can then use the value in the `Product` node to build the URL of the splash screen image to display. We will do this in the next part of the tutorial, Add a Control: Image[83].

## 4.1.6 Add a Control: Image

We will now add an image to the design. This image will be the splash screen of the Altova product that the end user selects in the combo box (*see previous section*[83]). The most important property of an image is the `Image URL` property, which selects the image to display. The URL we will build is a relative URL that looks for the image file in the same folder as the design file. You could, of course, use an absolute file URL or an image at an Internet location.

*In this part, you will:*

- Add an image control
- Define the `Image URL` property using an XPath expression that allows the URL to change dynamically with the combo box selection. The images referred to in this tutorial are located in the folder, `MobileTogetherDesignerExamples\Tutorials`[74], located in your *(*[72]*My) Documents folder*[72]. If you like you can save the images to another folder.

⊟ *Buttons in this section*

| | |
|---|---|
| X PATH | **XPath** |

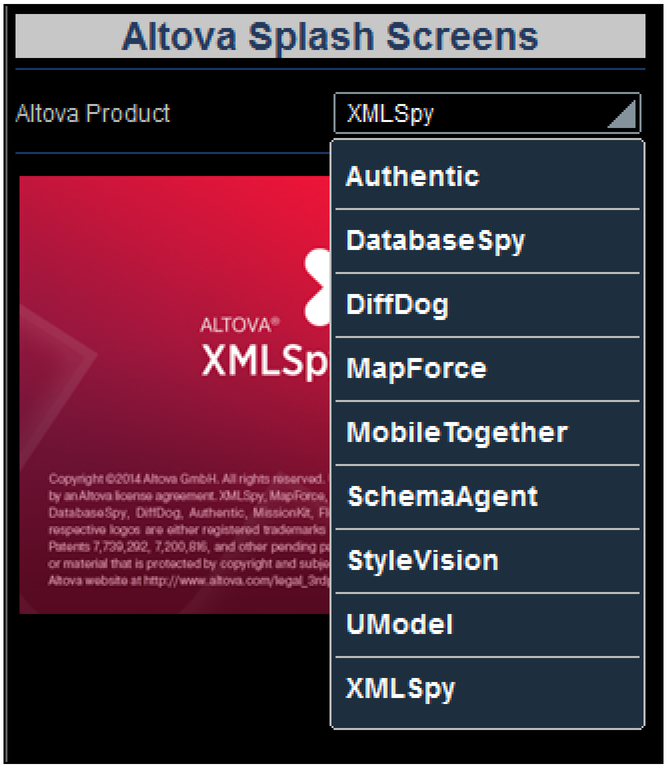
## Add an image

Add the image and define its properties as follows:

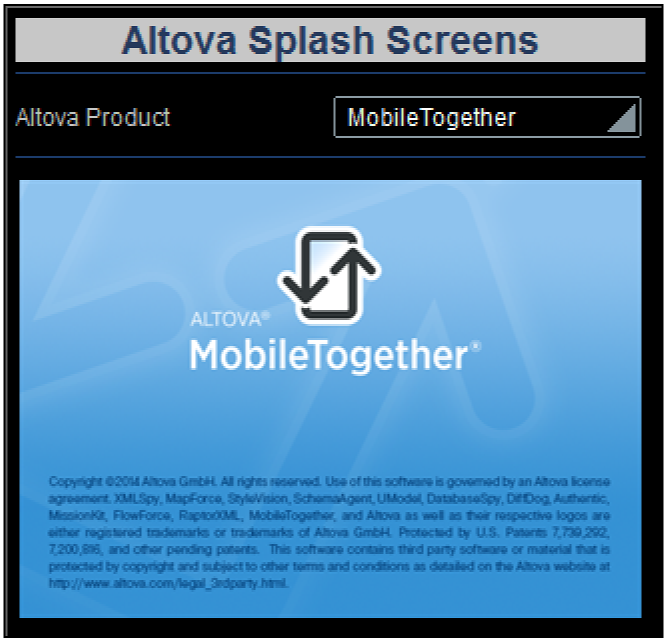
1. From the Controls Pane[266], drag and drop the Image control[541] below the table.
2. In the Styles & Properties Pane[274], name the image by setting its `Name` property to `Image: SplashScreen`.
3. Select the **`Image Source`** property, and click its **XPath** button or the **XPath** button in the toolbar of the Styles & Properties Pane[274]. In the Specify File dialog that appears, you can specify the path to the image file you want to display. Since we want to make this a dynamic path (that is, a screenshot that changes according to the product selected in the combo box), we will use an XPath expression to specify a dynamic filepath. To use an XPath expression, click the **XPath** button of the *Absolute/Relative Path* field. This displays the Edit XPath/XQuery Expression dialog[1244].
4. Enter the XPath expression: `concat(Product, '.bmp')`, and click **OK**.

This XPath expression produces a **relative URL** that locates a **`.bmp`** file in the same folder as the design file. You should specify the location that is correct for you, that is, the location where the image files have been saved. If you need to, you can use an absolute URL (*see example below*). The filename (for example, `xmlspy`) is obtained from the `Product` node, which in turn gets its content from the selection that the end user makes in the combo box. The default value of the `Product` node was set to `xmlspy`, so the XPath expression and starting image URL will be as below:

| | |
|---|---|
| *This XPath expression:* | **`concat(Product, '.bmp')`** |
| *Produces this absolute URL:* | `xmlspy`**`.bmp`** |



When the end user selects a product from the dropdown list of the combo box (*see screenshot above*), say `MobileTogether`, the splash screen of MobileTogether will be displayed (*screenshot below*). This is because the XML value corresponding to the MobileTogether selection is `mobiletogether`. This XML value is passed to the `Product` node, and it is used in the XPath expression to dynamically build the relative image URL: **`mobiletogether.bmp`**.



Copyright ©2014 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

To ensure that the image is reloaded whenever a combo box selection is made, a *Reload* action must be set on the combo box. How to do this is described in the next section, Define Control Actions 86.

## 4.1.7 Define Control Actions

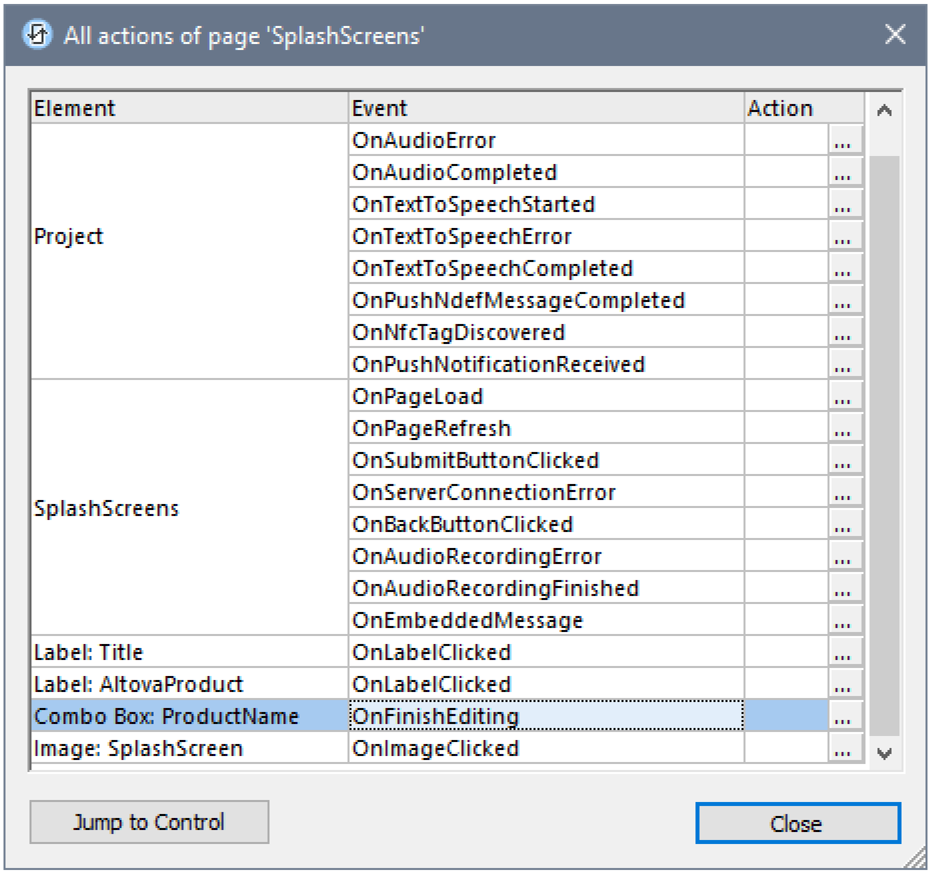
Control actions define what action a control takes in response to an event such as a button click or combo box selection. Actions that can be taken include: updating data nodes, reloading or saving page sources, or executing database commands.

*In this part, you will:*

- Look at all the page actions and control actions defined for the page
- Add a combo box action that updates the image whenever the combo box is edited

### Overview of page actions

To go to an overview of all actions of the `SplashScreens` page, click **Page | Actions Overview**. The Page Actions Overview dialog (*screenshot below*) appears. It displays all the events and their actions as currently defined for the page. The display includes page events and control events, and their respective actions. The `SplashScreens` element in the screenshot below refers to the page; all the other elements are the different controls of the page design. You can see that there is currently no action defined for any event.



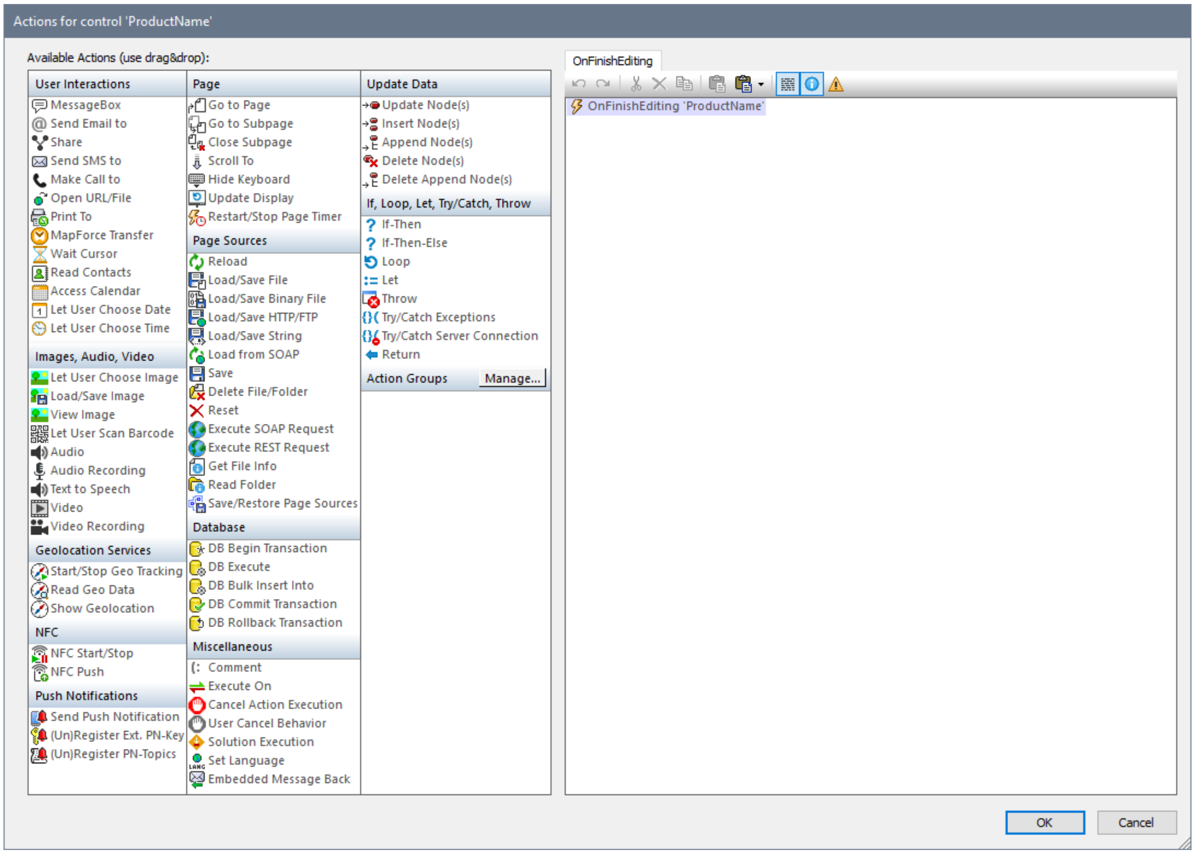
All actions of page 'SplashScreens'

| Element | Event | Action |
|---|---|---|
| Project | OnAudioError | ... |
|  | OnAudioCompleted | ... |
|  | OnTextToSpeechStarted | ... |
|  | OnTextToSpeechError | ... |
|  | OnTextToSpeechCompleted | ... |
|  | OnPushNdefMessageCompleted | ... |
|  | OnNfcTagDiscovered | ... |
|  | OnPushNotificationReceived | ... |
| SplashScreens | OnPageLoad | ... |
|  | OnPageRefresh | ... |
|  | OnSubmitButtonClicked | ... |
|  | OnServerConnectionError | ... |
|  | OnBackButtonClicked | ... |
|  | OnAudioRecordingError | ... |
|  | OnAudioRecordingFinished | ... |
|  | OnEmbeddedMessage | ... |
| Label: Title | OnLabelClicked | ... |
| Label: AltovaProduct | OnLabelClicked | ... |
| Combo Box: ProductName | OnFinishEditing | ... |
| Image: SplashScreen | OnImageClicked | ... |

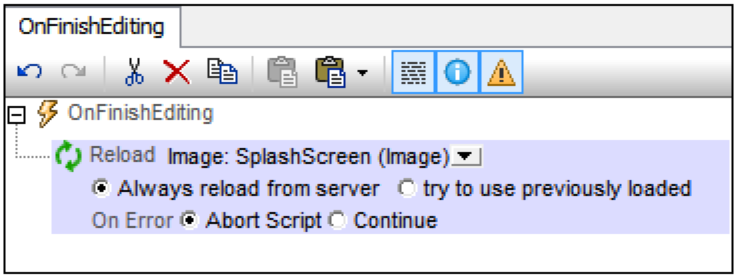
## Define the Reload action on the combo box control

Defining a control action consists of two parts: (i) selecting the control event that triggers the action; and (ii) specifying the action to perform when the event occurs. In our example, we want to do the following: When the end user makes a combo box selection, the image must be reloaded. This is because we want the image URL to be re-evaluated with the new value of the `Product` node (selected in the combo box). So, when combo box editing is finished, we want the *Reload* action to reload the image. Define this combo box event-and-action as follows.

1. Right-click the `XML:Product` combo box and select **Control Actions on OnFinishEditing**. This opens the Control Actions dialog for the combo box (*screenshot below*). There is only one combo box event: **`OnFinishEditing`** (*see the right-hand-side pane*). If additional events were available they would be shown as additional tabs of the right-hand pane. All the available actions for events are listed, in groups, in the left-hand-side pane.



2. Drag the `Reload` action from the Page Sources group and drop it in the **`OnFinishEditing`** tab (*screenshot below*). This specifies that a reload action must be carried out when the combo box has been edited.
3. Click the drop-down arrow (next to `$XML1`), select `Image: SplashScreen`, and click **OK**. This specifies that the image control will be updated when the combo box value is changed.



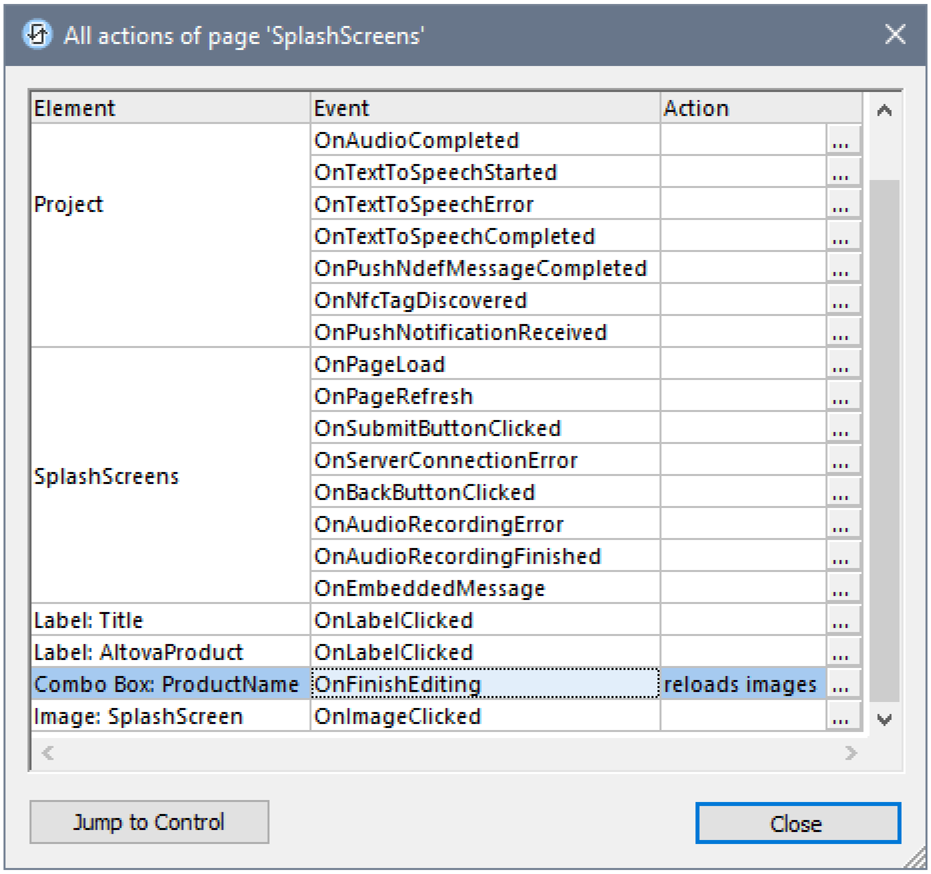
OnFinishEditing

Reload Image: SplashScreen (Image)

● Always reload from server ○ try to use previously loaded

On Error ● Abort Script ○ Continue

If you now check the Page Actions Overview dialog, you will see that a *Reloads Image* action is defined for the combo box's `OnFinishEditing` event.



| Element | Event | Action |
|---|---|---|
|  | OnAudioCompleted |  |
|  | OnTextToSpeechStarted |  |
| Project | OnTextToSpeechError |  |
|  | OnTextToSpeechCompleted |  |
|  | OnPushNdefMessageCompleted |  |
|  | OnNfcTagDiscovered |  |
|  | OnPushNotificationReceived |  |
| SplashScreens | OnPageLoad |  |
|  | OnPageRefresh |  |
|  | OnSubmitButtonClicked |  |
|  | OnServerConnectionError |  |
|  | OnBackButtonClicked |  |
|  | OnAudioRecordingError |  |
|  | OnAudioRecordingFinished |  |
|  | OnEmbeddedMessage |  |
| Label: Title | OnLabelClicked |  |
| Label: AltovaProduct | OnLabelClicked |  |
| Combo Box: ProductName | OnFinishEditing | reloads images |
| Image: SplashScreen | OnImageClicked |  |

## 4.1.8 Validate the Project

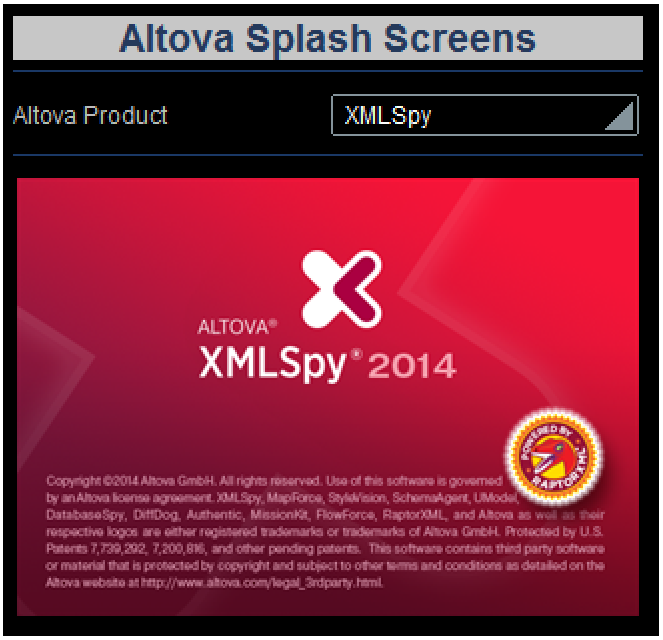
Now that the design of the page is complete, you should validate the project (**Project | Validate**) to check for errors. On validating, you will get a message saying that there are no errors or warnings.

- Warnings indicate design issues that might need your attention, but these issues are not fatal and will not prevent the solution from running.
- An error message, on the other hand—as opposed to a warning—would be fatal and should be remedied.

**Note:** A number of messages in the Messages window can be expanded to reveal more detail, and often contain clickable links that provide more information.

## 4.1.9 Run a Simulation

You can run a simulation of the project workflow directly within MobileTogether Designer. The simulation device will be the device currently selected in the preview device combo box[74] of the Main toolbar. You can change the preview device to see the simulation on different devices. To run the simulation, select **Run | Simulate Workflow** or **F5**. This opens the simulator and starts the simulation.

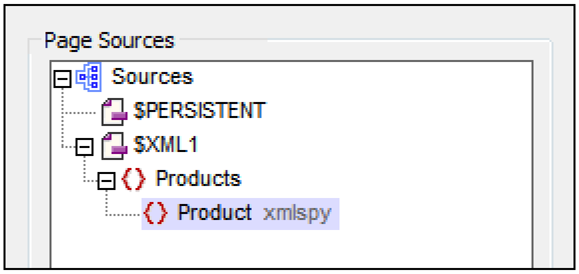


Altova Splash Screens

Altova Product: XMLSpy

Copyright ©2014 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

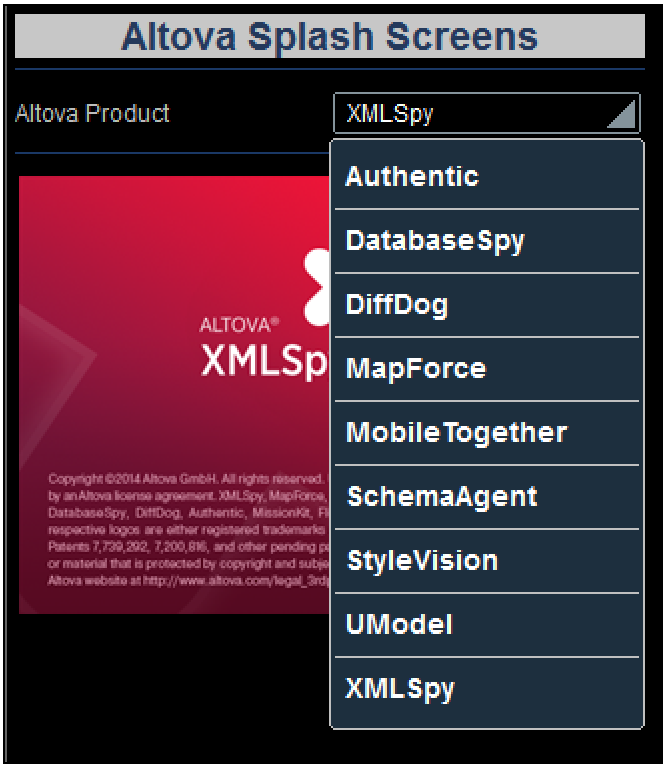
The XMLSpy splash screen is displayed because the default value of `Product` was set to `xmlspy`. You can see this in the *Page Sources* pane on the right if you expand the `$XML1` node (*screenshot below*).



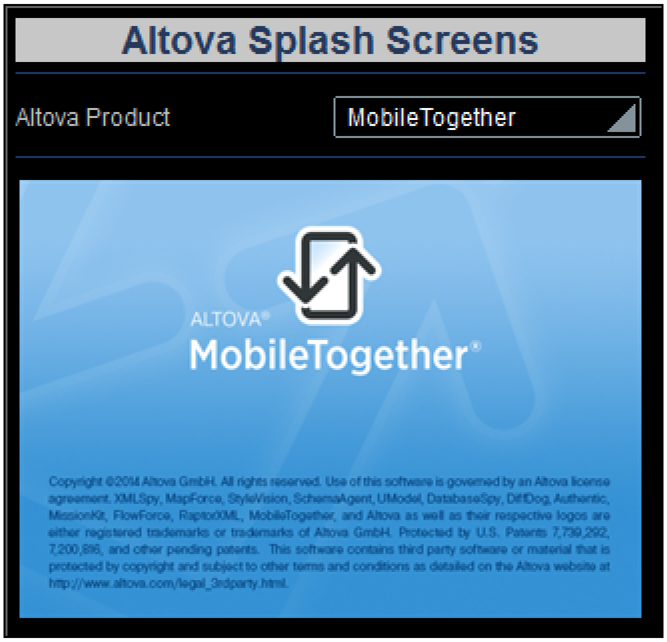
Page Sources

```
Sources
  $PERSISTENT
  $XML1
    Products
      Product xmlspy
```

Now, in the simulation, click the combo box to display its dropdown list (*screenshot below*).


Altova Splash Screens
Altova Product
XMLSpy
Authentic
DatabaseSpy
DiffDog
MapForce
MobileTogether
SchemaAgent
StyleVision
UModel
XMLSpy
ALTOVA®
XMLSp
Copyright ©2014 Altova GmbH. All rights reserved.
by an Altova license agreement. XMLSpy, MapForce,
DatabaseSpy, DiffDog, Authentic, MissionKit, F
respective logos are either registered trademarks
Patents 7,739,292, 7,200,816, and other pending p
or material that is protected by copyright and subje
Altova website at http://www.altova.com/legal_3rd

Select a product from the list and notice how the image and value of the `Product` node changes. (Note that, for iOS devices, you will need to press **Set** to confirm selections.)

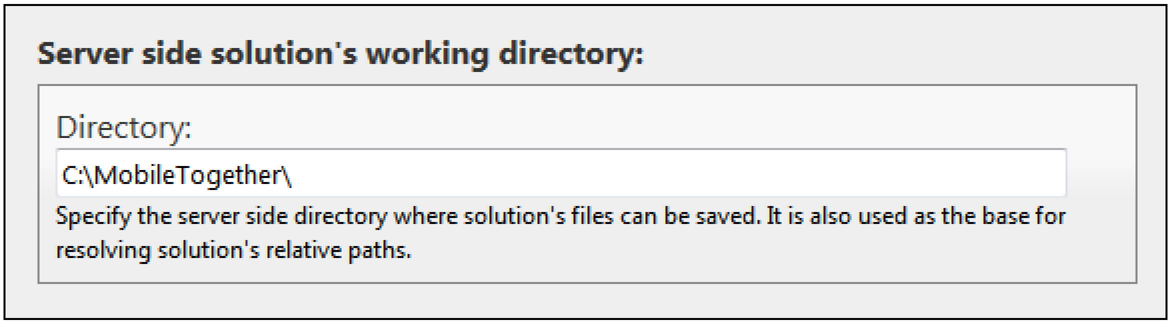


Copyright ©2014 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

After you have finished, click **Close** or press **Esc** to close the simulator.

### Running a server simulation

A server simulation uses MobileTogether Server to run the simulation (**Run | Use Server for Workflow Simulation**). In the Web UI of MobileTogether Server, you must set the solution's working directory (**Settings | Server Side Solution's Working Directory**, *see screenshot below*). All relative paths in the design will be resolved relative to the directory specified in this setting. In order for server simulation to work correctly, enter the path of the directory where your referenced files are saved. The directory setting shown in the screenshot (`C:\MobileTogether\`) means that both the design and image files must be in the folder `C:\MobileTogether\` (since the relative image URLs in the design indicate that the image files are to be found in the same directory as the design).



Server side solution's working directory:

Directory:

C:\MobileTogether\

Specify the server side directory where solution's files can be saved. It is also used as the base for resolving solution's relative paths.

## 4.1.10 Deploy to Server

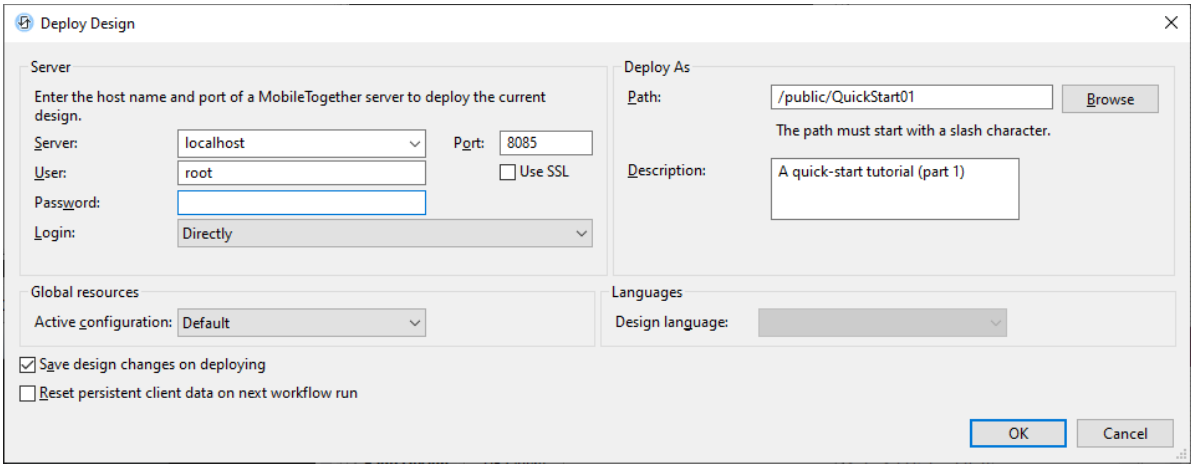
After you have successfully validated the project and tested the simulations, you are ready to deploy the project to MobileTogether Server. When the project is deployed to the server, it becomes a solution that a MobileTogether Client app can run. For more information about deploying the project and files, see the sections Deploying the Project[291] and Location of Project Files[289].

In order to deploy the project to MobileTogether Server, you will need to access the server as a user having the privilege, *Save Workflow from Designer*. The access rights of users of MobileTogether Server are defined in the Web UI of MobileTogether Server. See the MobileTogether Server user manual for information about setting user privileges.

### Deploying the project to MobileTogether Server

Deploy the project and its associated (image) files to MobileTogether Server as follows:

1. In the Files Pane[259] (*screenshot below*), right-click Image Files, and, in the context menu that appears, select **Add Deployable File**.
2. Browse for the image file and add it to the Image Files list. You can select multiple files if you like. Click **Open**.
3. In the dialog that appears, choose whether you want the selected files to be deployed to the server, the client, or both, then click **Yes**. You will be prompted for each file separately.
4. The selected files will be added. Each file will be shown with its relative path and its deployment location (server, client, or both). Both properties can be changed in the context menu of each file.
5. Select **File | Deploy to MobileTogether Server**. This displays the Deploy Design dialog (*screenshot below*).



Deploy Design

Server

Enter the host name and port of a MobileTogether server to deploy the current design.

Server: localhost

Port: 8085

User: root

□ Use SSL

Password:

Login: Directly

Deploy As

Path: /public/QuickStart01

The path must start with a slash character.

Description: A quick-start tutorial (part 1)

Global resources

Active configuration: Default

Languages

Design language:

☑ Save design changes on deploying

□ Reset persistent client data on next workflow run

6. Enter the MobileTogether Server address and administrator port (default `HTTP` port is `8085`, `HTTPS` is `8086`).
7. Enter the user name and password with which you want to access MobileTogether Server. Note that this user must have the *Save Workflow from Designer* privilege in order for deployment to be successful. See the MobileTogether Server user manual for information about user privileges.
8. In the *Deploy As* | *Path* field, enter the name under which you wish to deploy this solution on the server. For example, in the screenshot above, the solution will be saved in the **`/public`** container and will have the name **`QuickStart01`**. On MobileTogether Server, you will see only the one entry: **`QuickStart01`**. The image files will be contained in this entity and will not be visible as separate files.
9. Click **OK** when done. The project and the files selected for deployment are now deployed to MobileTogether Server as the workflow, **`QuickStart01`**.

That's it!

# 4.2 QuickStart (Part 2)

The second part of this QuickStart tutorial continues where Part 1 left off. It focuses on using an external XML file as its page source. You will learn how to generate a dynamic dropdown list for the combo box from the data in the XML file, how to create dynamic links to website pages, and how to save data back to the XML file.

This tutorial is organized into the following sections, each of which deals with one key aspect of creating a MobileTogether design.

- Load Data from a File 94
- Change Source Node 97
- Run a Simulation 98
- Use File Data for Combo Box Entries 100
- Set Data File as Default File 102
- Create Dynamic Links to Web Pages 104
- Save Data Back to File 105

## The tutorial files

The files for this tutorial are located in your ( 72 *My) Documents* 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\QuickStart`**. The `Tutorials` folder also contains the splash screen images referred to in the tutorial. You can save these images and the XML file to any other location and use them from there if you like.

- The file to start with is the file you created in Part 1. Alternatively, you can use the supplied design file, `QuickStart01.mtd`
- The file you will end with should be similar to `QuickStart02.mtd`
- The XML data file is `AltovaProducts.xml`
- The image files are: `*.bmp`

## Video demo of how to build your first app

The Altova website provides a video demo that describes the user interface and explains the first steps to take when creating a design.

## 4.2.1 Load Data from a File

*In this part, you will:*

- Specify that the data for the solution be taken from an XML file when the solution page is loaded. The file, located in the Tutorials folder 72, is called `AltovaProducts.xml` and its listing is given below. It has a similar structure to the page source created in Part 1 77, with the only difference being that there is one new element: `Selection`.
- Modify the tree structure of the page source to match the different structure of the XML data file.

⊟ *Listing of the XML file, AltovaProducts.xml*

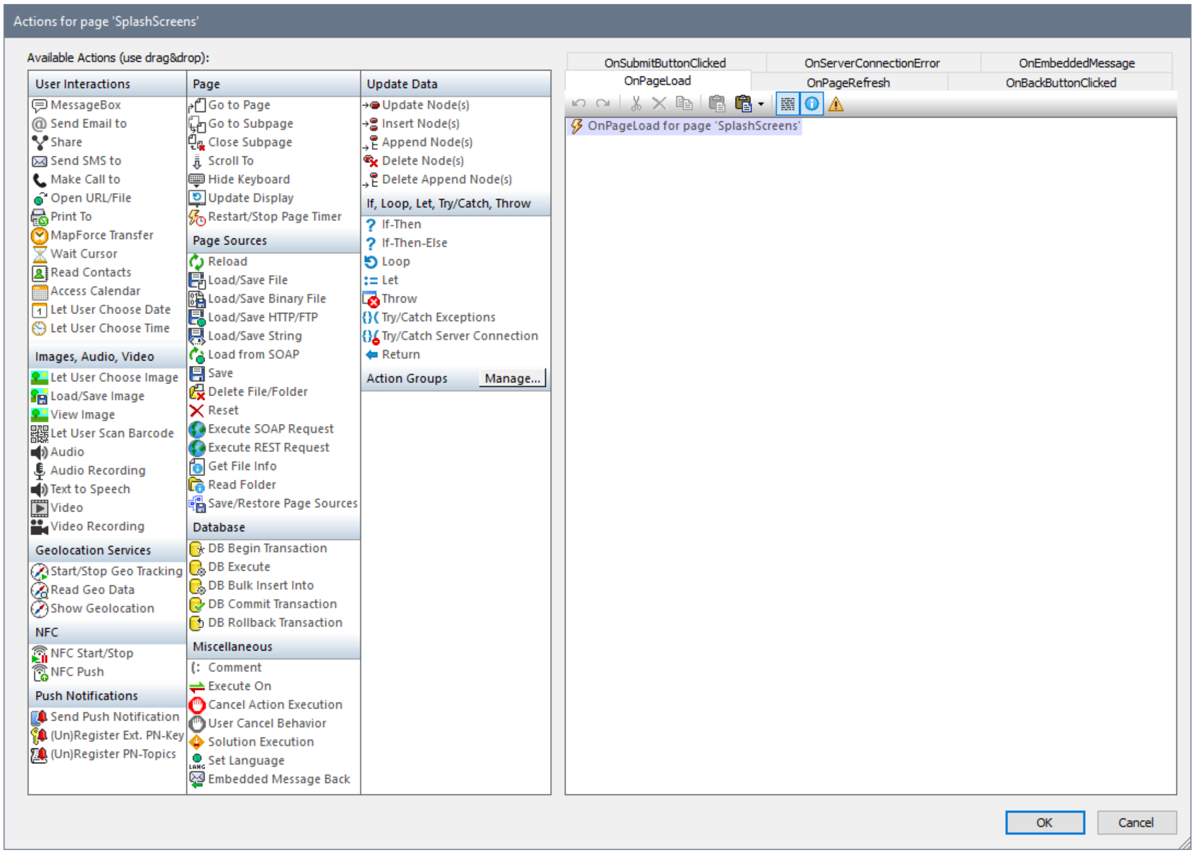
Located in the MobileTogether folder of the (72 My) Documents folder 72: **`MobileTogetherDesignerExamples\Tutorials\AltovaProducts.xml`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<Products>
<Selection></Selection>
<Product>XMLSpy</Product>
<Product>MapForce</Product>
<Product>StyleVision</Product>
<Product>MobileTogether</Product>
<Product>DatabaseSpy</Product>
<Product>DiffDog</Product>
<Product>SchemaAgent</Product>
<Product>UModel</Product>
<Product>Authentic</Product>
</Products>
```

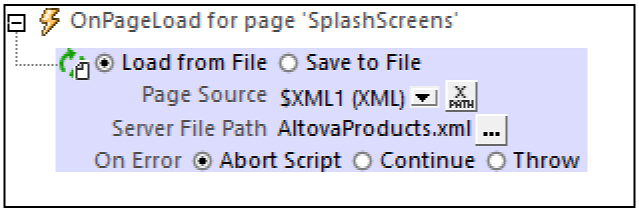
## Specify data file to use on page load

To specify that data for the page source be taken from an XML file, do the following:

1. Open `QuickStart01.mtd`, which is located in the MobileTogether folder of the (72 My) Documents folder 72: **`MobileTogetherDesignerExamples\Tutorials\`**.
2. Click **Page | Page Actions**. This displays the Page Actions dialog (*screenshot below*).



3. Drag and drop the *Load/Save File* action into the tab of the `OnPageLoad` event.
4. Make sure that the Load from File option is selected (*see screenshot below*), and that the page source is **`$XML1`**.
5. Click the **Additional Dialog** button of the **`File Path`** entry. This displays the Load File dialog.
6. Select *Make path relative to design file* and browse for the file, `AltovaProducts.xml`.
7. You will be asked whether you want to deploy this file together with the design file to MobileTogether Server. Click **Yes**. The file will be set as the data file to load for the page source **`$XML1`** when the page is loaded (*screenshot below*).



OnPageLoad for page 'SplashScreens'

● Load from File ○ Save to File

Page Source $XML1 (XML)

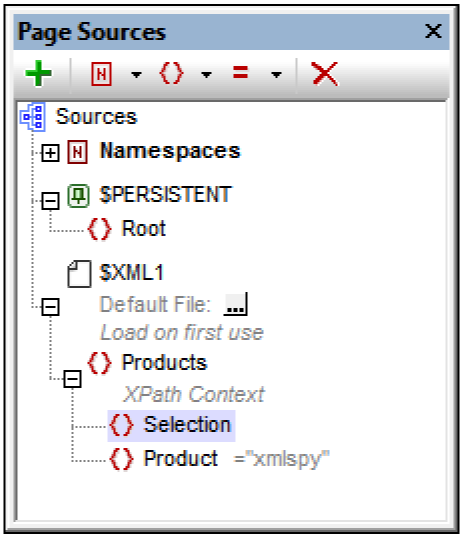
Server File Path AltovaProducts.xml ...

On Error ● Abort Script ○ Continue ○ Throw

8. Click **OK** to finish.

## Modify the data structure of the page source

The XML data file has an extra `Selection` element. So, in order for the XML tree to hold the data from this element, we will now add a `Selection` element to the XML tree of the page source in the Page Sources Pane[270] (*see screenshot below and listing above*[94]). Add the `Selection` element to the tree by right-clicking either `Products` or `Product` and selecting, respectively, **Add Child** or **Append**, and then **Element**. Rename the element to `Selection` by double-clicking the element and then editing the name.



We will not add any default value for `Selection`. This is because, when the page is loaded, we want the data for the page to come from the file `AltovaProducts.xml`. It was the action we defined for the `OnPageLoad` event of this page (*see above*). If we were to set a default value for `Selection`, then this default value would override the value obtained from the `Selection` node in `AltovaProducts.xml`. So with no default values assigned in the Page Sources Pane, when the page is loaded, the `Selection` node will be empty. This is because the `Selection` node in `AltovaProducts.xml` is empty (*see the file listing above*). We will test this in simulations later in the tutorial.

## 4.2.2 Change Source Node

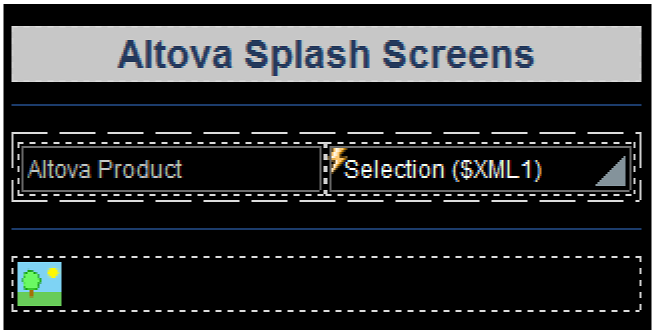
*In this part, you will:*

- Change the source node of the combo box
- Save the XPath expression of the image

### Change source node of the combo box

Each page control can have a **source node (or page source link)**, which is a node in one of the page sources. The link is made by dragging the page source node from the Page Sources Pane[270] onto the control in the design. Essentially, a source node transfers data from the XML node to the control. But how the data in the XML node relates exactly to the control depends on the type of control it is. For example: A combo box selection updates its page source link—an XML node—and that value is reflected in the combo box display, whereas the source node of an image provides the URL of the image. When you hover over a control, the popup indicates how the source node will be used, for example, as an XML node to edit (for combo boxes), or as a data originator (for images).

We will change the source node of the combo box, from **`Product`** to **`Selection`**. Do this by dragging the **`Selection`** node from the Page Sources Pane[270] onto the combo box control (*see screenshot below*).



We do this because we want to put the end user's combo box selection in the **`Selection`** element rather than the **`Product`** element. The reasons for wanting this are:

- There are multiple sibling **`Product`** elements in the file `AltovaProducts.xml`, each of which contains data we do not want to change.
- If the source node were `Product`, only the first `Product` element (`Product[1]`) would be updated with the combo box selection. This is undesirable.
- The best solution would be to store the end user selection in a separate element.

After changing the source node from **`Product`** to **`Selection`**, the combo box selection will update the **`Selection`** node—and not the **`Product`** node.

### Change XPath expression of the image URL

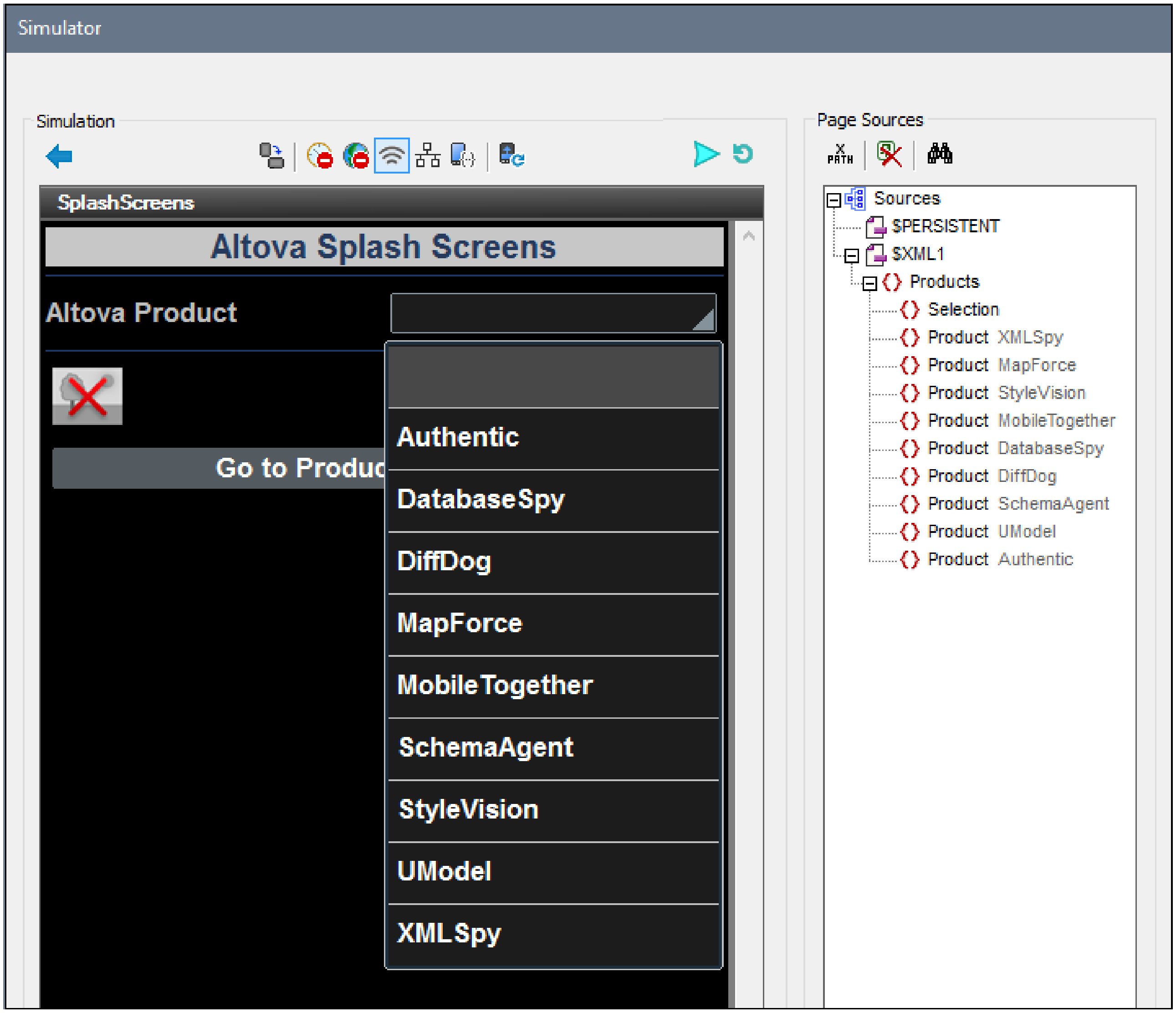
Since the XML value of the combo box selection goes into the **`Selection`** node, this node must be used in the XPath expression that constructs the image URL. In the design, select the image, and click the **XPath** button of the **`Image URL`**[541] property (in the Styles & Properties Pane[274]). In the Specify File dialog that appears, modify the XPath expression so that `Product` is replaced by `Selection`. For example:

*If you have:* `concat(Product, '.bmp')`
*Change it to:* `concat(Selection, '.bmp')`

This XPath expression uses the end user's combo box selection (now stored in the **`Selection`** node) to generate the image file name. Since the image file and design file are in the same folder, the file name generated by the XPath expression is also the relative path to the image file from the location of the design file.

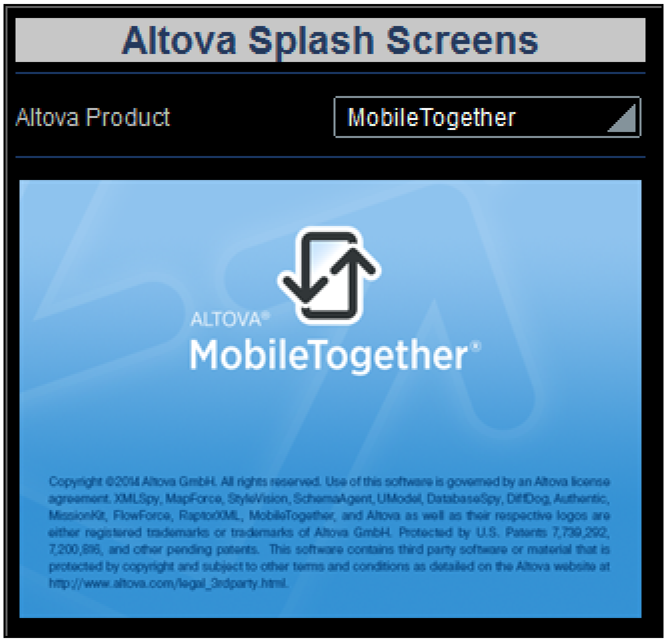
## 4.2.3 Run a Simulation

If you run a simulation now (**Run | Simulate Workflow** or **F5**), you will see the following:



| What you see | Reason |
|---|---|
| **`Selection`** node in XML Data tree is empty | Value comes from the empty **`Selection`** node in `AltovaProducts.xml`, [the file that is loaded on page open](94) |
| Combo box is empty | Because the **`Selection`** node is empty |
| Dropdown list has empty entry | Empty entry added as a result of current combo box selection (=`empty`) |
| No splash screen is displayed | [Image URL is built](97) using the empty **`Selection`** node |

If you now select a product from the dropdown list (for example, `MobileTogether`), the splash screen of that product will be displayed (*screenshot below*).



Copyright ©2014 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

This is because:

1. A combo box selection (of, say, **`MobileTogether`**) puts the corresponding value (`mobiletogether`) into the **`Selection`** node.
2. The value in the **`Selection`** node is used to [correctly construct the image URL](97).

Notice also that the combo box, from now onwards, no longer has the empty entry in its dropdown list. This is because the **`Selection`** node is not empty any longer and because the list of defined combo box entries, therefore, does not have an empty entry.

After you have finished, click **Close** or press **Esc** to close the simulator.

## 4.2.4 Use File Data for Combo Box Entries

*In this part, you will:*

- Use the data tree structure to generate the combo box entries
- Run a simulation to test the effect of the change

⊟ *Listing of the XML file, AltovaProducts.xml*

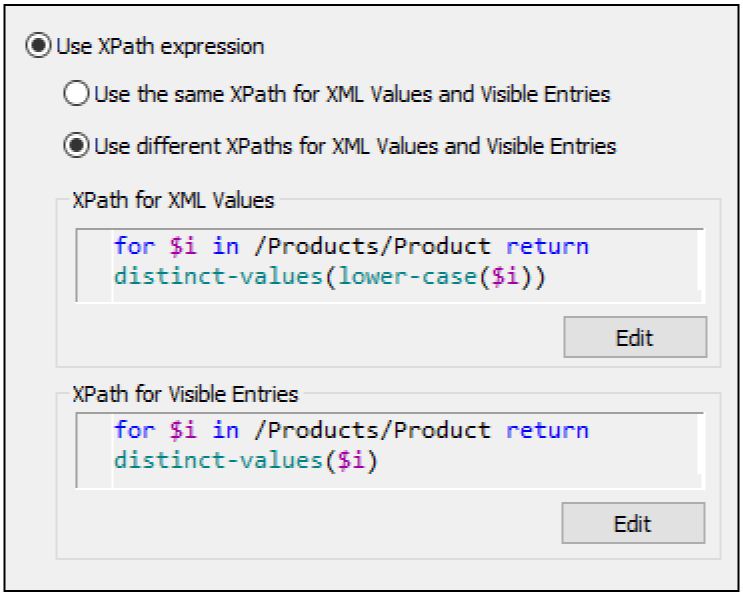
Located in MobileTogether folder of *([72]My) Documents folder*[72]: **`MobileTogetherDesignerExamples\Tutorials\AltovaProducts.xml`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<Products>
<Selection></Selection>
<Product>XMLSpy</Product>
<Product>MapForce</Product>
<Product>StyleVision</Product>
<Product>MobileTogether</Product>
<Product>DatabaseSpy</Product>
<Product>DiffDog</Product>
<Product>SchemaAgent</Product>
<Product>UModel</Product>
<Product>Authentic</Product>
</Products>
```

### Edit the combo box entries

Edit the combo box entries as follows:

1. Select the combo box, and, in the Styles & Properties Pane[274], click the Additional Dialog button of the **`Combo Box Entry Values`** property. The Edit Combo Box dialog (*screenshot below*) appears.



● Use XPath expression

○ Use the same XPath for XML Values and Visible Entries

● Use different XPaths for XML Values and Visible Entries

XPath for XML Values

```
for $i in /Products/Product return
distinct-values(lower-case($i))
```

XPath for Visible Entries

```
for $i in /Products/Product return
distinct-values($i)
```

2. Select *Use XPath expression*, and then *Use different XPath for XML Values and Visible Entries*.
3. Enter the XPath expressions for *Visible Entries* and *XML Values* as shown in the screenshot above.
4. Select the *Sort values* check box at the bottom of the dialog to sort the list when it is displayed.
5. Click **OK** to finish.

Remember that the **`Products`** node was defined as the default XPath context node for this page 79. The XPath **`for`** expression iterates over the `Product` child nodes of `Products` (the context node) and returns a sequence of all the unique distinct values, alphabetically sorted. In the case of the *XML Values* sequence, the values are transformed to lowercase before being filtered for uniqueness. These two sequences are the entries of the dropdown list (*Visible Entries*) and their corresponding XML values (*XML Values*). The advantage of using the data tree structure to build the combo box entries and a data source file to load data is that combo box entries are generated dynamically from the data source file; they are not hard-coded as items of a list in the design. Consequently, if a new product is added to the XML file, it will automatically appear as an entry in the dropdown list.

## Run a simulation

When you run a simulation, it will run in exactly the same way as when the combo box entries were defined as a list (*see the previous section*, Run a Simulation 98). The only difference will be that the entries of the dropdown list will be the values of the `Product` elements in **`AltovaProducts.xml`** (*see listing above*). When an entry from the dropdown list is selected, the corresponding (lowercase) XML value will be entered in the **`Selection`** node, and the image URL will be correctly evaluated 97.

## Change data in the data source file

Make the following two changes in the data source file, `AltovaProducts.xml` (*listing above*):

- Add a lowercase product name to the `Selection` node as shown in the listing below
- Remove a few `Product` elements from the file, or add some `Product` elements to the file, and change the order of the `Product` elements. In the Edit Combo Box dialog (*see above*), also test the effect of selecting/deselecting the *Sort values* check box.

```
<?xml version="1.0" encoding="UTF-8"?>
<Products>
  <Selection>databasespy</Selection>
  <Product>XMLSpy</Product>
    ...
    ...
  <Product>DatabaseSpy</Product>
</Products>
```

After making these changes, save the file and run a simulation. The initial splash screen will be that of the product in the `Selection` node. Also, the dropdown list of the combo box will not have any empty entry, and the number of entries in the dropdown list will be equal to the number of unique `Product` elements in the XML file.

## 4.2.5 Set Data File as Default File

*In this part, you will:*

- Specify a default file for the page source
- Run a simulation

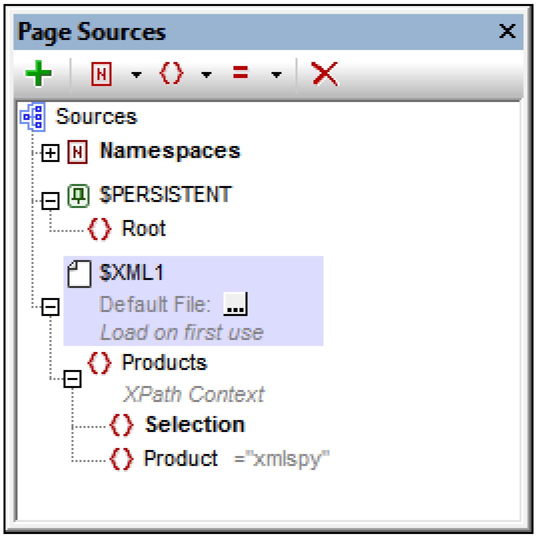
⊟ *Buttons in this section*

[...] **Additional Dialog**

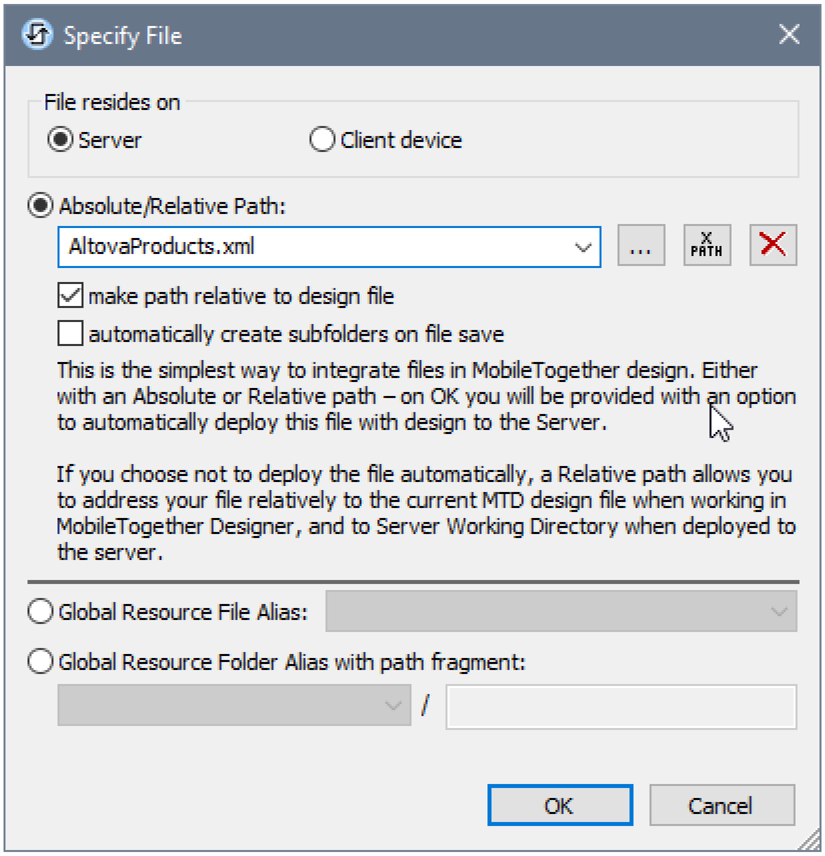
### Specify a default file for the page source

The data that goes into a page source can be specified by selecting a default file for the page source. Do this as follows:

1. Click the **Additional Dialog** button of the **`$XML1`** default file (*screenshot below*).



2. Select *Server* and *Absolute/Relative Path*, check the *Make path relative to design file* check box, and then browse for the file **AltovaProducts.xml** 94.



Specify File

File resides on

● Server ○ Client device

● Absolute/Relative Path:

AltovaProducts.xml

☑ make path relative to design file

□ automatically create subfolders on file save

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:

○ Global Resource Folder Alias with path fragment:

3. On clicking **OK**, the file will be added as the default file, and its data will populate the page source tree.
4. Click **Page | Page Actions** to open the Page Actions dialog.

5. In the `OnPageLoad` tab, select the Load from File entry[94] and delete it. This is because the *Load from File* action is now redundant since the file `AltovaProducts.xml` has been specified as the default file of the page's only page source.
6. Run a simulation to test whether the default file is being used.

## 4.2.6 Create Dynamic Links to Web Pages

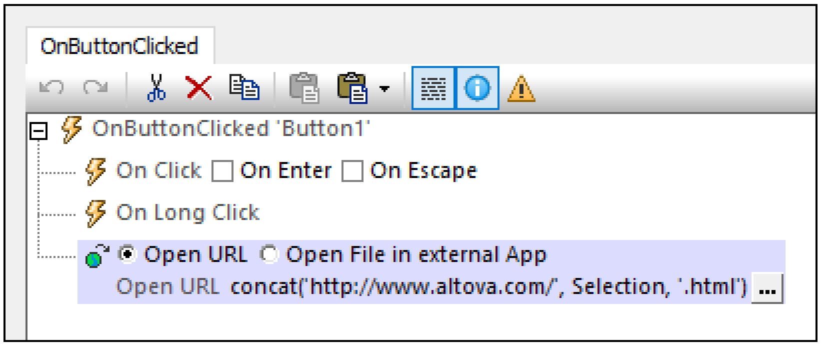
*In this part, you will:*

- Add a button that links dynamically to a web page (using an XPath expression)
- Run a simulation

### Add a button that links to a web page

We will now add a button that links to the product description page of the product selected in the combo box. Do this as follows:

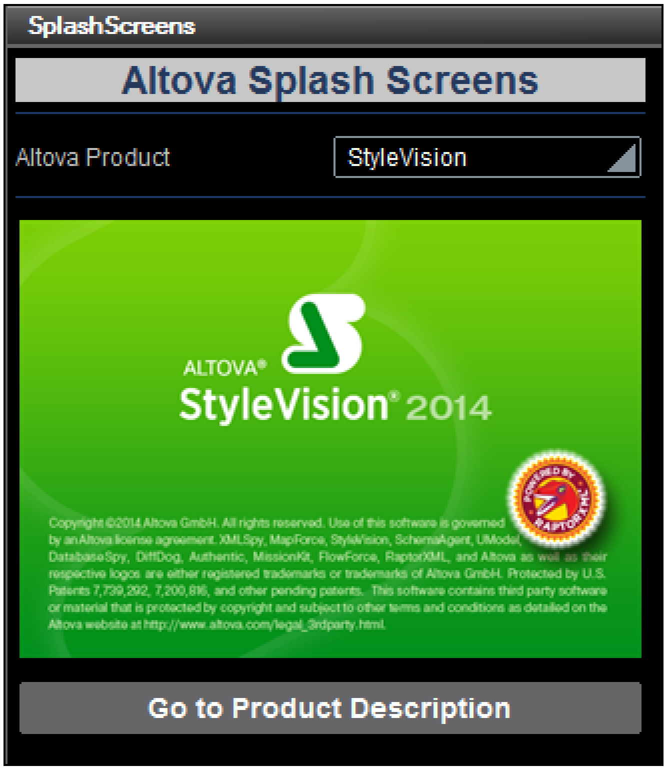
1. Drag the button control from the Controls Pane[266] and drop it below the image (*see the simulator screenshot below*).
2. Enter the text *Go to Product Description*.
3. Right-click the button and select **Control Actions for OnButtonClicked**.
4. In the Actions dialog that appears (*screenshot below*), drag the *Open URL/File* action into the `OnButtonClicked` tab and drop it below the `OnClick` and `OnLongClick` events as shown in the screenshot below. Since there is no action specified for **either type of click**, the *Open URL* action is performed as the next (additional) action to perform after any of the two events is triggered[417].
5. Click the **XPath** button, and, in the Edit XPath/XQuery Expression dialog that appears, enter the XPath expression: `concat('http://www.altova.com/', Selection, '.html')`


OnButtonClicked
OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
● Open URL ○ Open File in external App
Open URL concat('http://www.altova.com/', Selection, '.html')
...

6. Click **OK** to finish, and save the file.

### Run a simulation

Run a simulation by clicking **F5** or **Run | Simulate Workflow**. When the simulation starts, select a product in the combo box and then click the **Go to Product Description** button (*see screenshot below*). This will take you to the product description page on the Altova website.



Copyright ©2014 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

## 4.2.7 Save Data Back to File

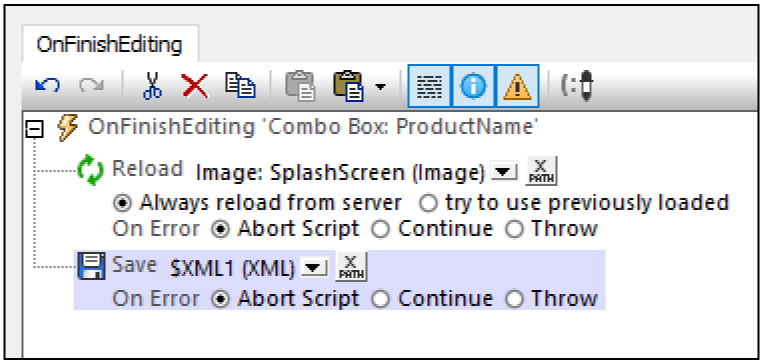
*In this part, you will:*

- Save changed data back to file using a control action
- Run a simulation to test the success of the *Save to File* action

### Save data to file after editing combo box

You can specify that a change made by editing the combo box is saved back to file. Since the source node of the combo box is the `Selection` element, any change made to the combo box selection will be saved to this element. To specify that any change is saved back to the `Selection` element in the default file, we will add the *Save* action to the **`OnFinishEditing`** event of the combo box. Do this as follows:

1. Right-click the combo box and select **Control Actions for OnFinishEditing** from the context menu.
2. This displays the Control Actions dialog for the combo box, which already has the *Reload* action that targets the image.
3. Drag the *Save* action into the *OnFinishEditing* pane, and drop it below the *Reload* action (*see screenshot below*).
4. Open the action's Edit XPath Expression dialog, and enter `$XML1` as the source to which the data should be saved (*see screenshot below*).



OnFinishEditing 'Combo Box: ProductName'

Reload Image: SplashScreen (Image)

● Always reload from server ○ try to use previously loaded

On Error ● Abort Script ○ Continue ○ Throw

Save $XML1 (XML)

On Error ● Abort Script ○ Continue ○ Throw

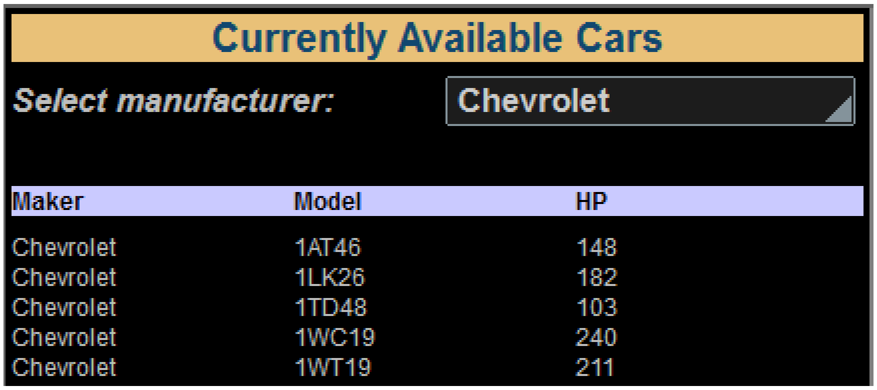
5. Click **OK** to finish, and then save the file.
6. To test whether the change is saved to the default file, open the default file in an editor, run a simulation, select a combo box entry, and then reload the default file in the editor. The new combo box selection will appear as content of the `Selection` element in the default file.

**Note:** If you wish to save to some file other than the default file, use the *Save to File* action (instead of the *Save* action). If you wish to save to a web page, use the *Save to HTTP/FTP action*. (In this case, you will also need to provide the authentication details that allow the user to modify the page at the HTTP URL.)

That's it!

# 4.3 Simple Database

This tutorial shows how to: (i) create a simple design based on a DB data source, and (ii) load and display DB records on the basis of a user selection. The data for the design is sourced from a database of cars that is stored in a Microsoft Access database. In the solution, the user can select a manufacturer. That manufacture's car models are then displayed in a table (*see screenshot below*).



Currently Available Cars

Select manufacturer: Chevrolet

| Maker | Model | HP |
|---|---|---|
| Chevrolet | 1AT46 | 148 |
| Chevrolet | 1LK26 | 182 |
| Chevrolet | 1TD48 | 103 |
| Chevrolet | 1WC19 | 240 |
| Chevrolet | 1WT19 | 211 |

## The tutorial files

The files for this tutorial are located in your ([72]My) Documents[72] MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\Databases`**.

- The Access database that contains records of the models of cars made by some manufacturers: `MyCars.mdb`
- The design file you will end with should be similar to `SimpleDatabase.mtd`

## Tutorial structure

This tutorial is organized into the following sections:

- The DB Data Source[107]
- Persistent Tree for User Input[110]
- Load DB Data Based on User Selection[112]

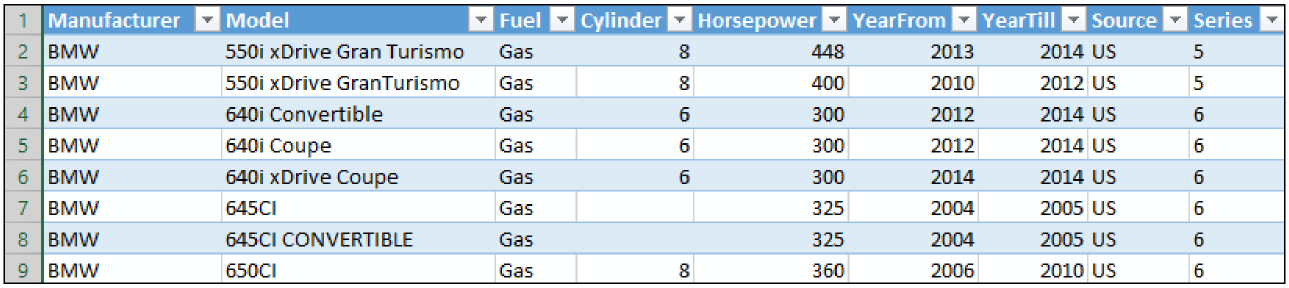
## Video demo of how to build a database-driven app

The Altova website provides a video demo that shows how to connect to a DB and query it, how to retrieve DB records, and how to present DB data in the form of tables.

### 4.3.1 The DB Data Source

In this section, a new MobileTogether design is created. It uses the Microsoft Access database `MyCars.mdb` as its data source.

The structure of the DB is as shown in the screenshot below. (Note that the screenshot shows only the first few records of the DB.)

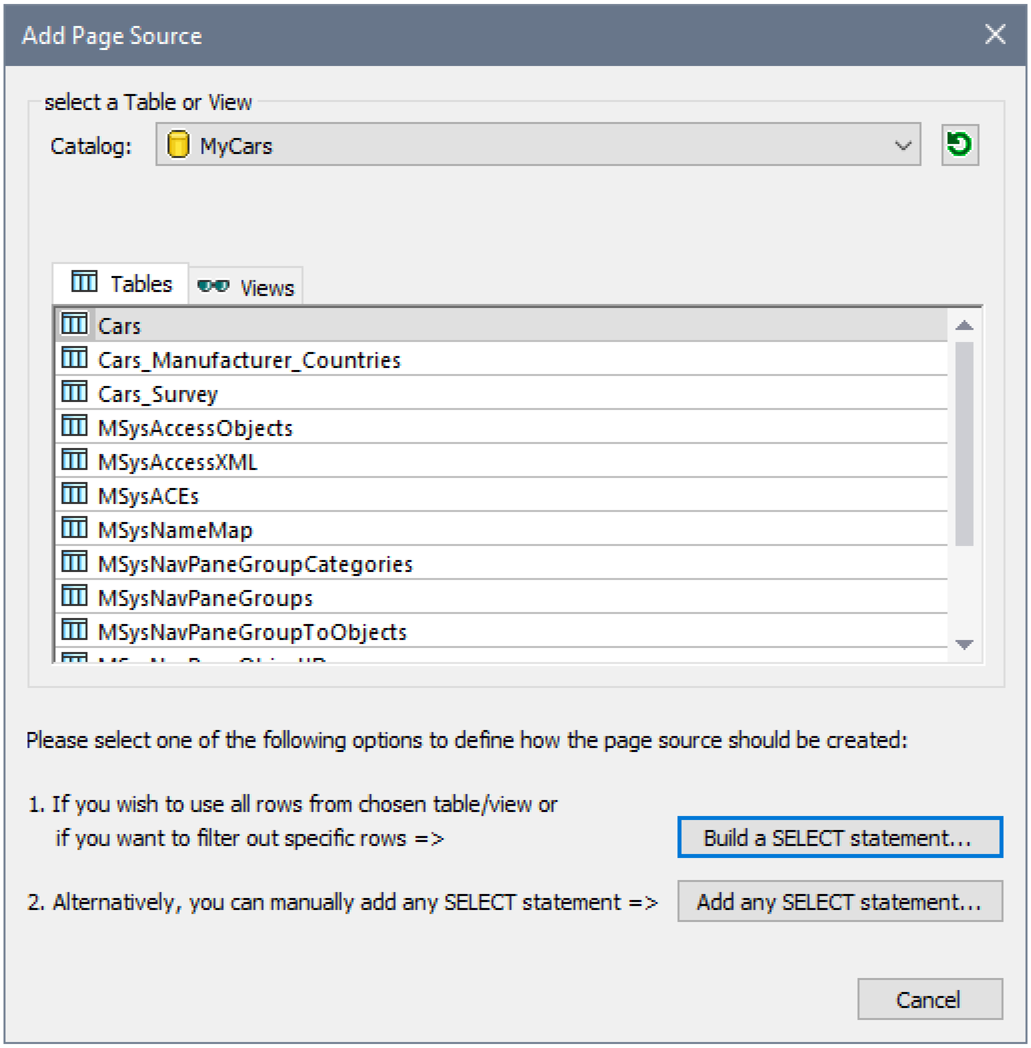


| 1 | Manufacturer | Model | Fuel | Cylinder | Horsepower | YearFrom | YearTill | Source | Series |
|---|---|---|---|---|---|---|---|---|---|
| 2 | BMW | 550i xDrive Gran Turismo | Gas | 8 | 448 | 2013 | 2014 | US | 5 |
| 3 | BMW | 550i xDrive GranTurismo | Gas | 8 | 400 | 2010 | 2012 | US | 5 |
| 4 | BMW | 640i Convertible | Gas | 6 | 300 | 2012 | 2014 | US | 6 |
| 5 | BMW | 640i Coupe | Gas | 6 | 300 | 2012 | 2014 | US | 6 |
| 6 | BMW | 640i xDrive Coupe | Gas | 6 | 300 | 2014 | 2014 | US | 6 |
| 7 | BMW | 645CI | Gas |  | 325 | 2004 | 2005 | US | 6 |
| 8 | BMW | 645CI CONVERTIBLE | Gas |  | 325 | 2004 | 2005 | US | 6 |
| 9 | BMW | 650CI | Gas | 8 | 360 | 2006 | 2010 | US | 6 |

## Set up the DB data source in the design

Set up the DB data source as follows:

1. Create a new design (**File | New**).
2. Create a page source by clicking the **Add Source** icon in the Page Sources pane[270].
3. In the Add Page Source dialog[317] that appears, select *New DB Structure* and then **Next**.
4. In the next screen of the dialog, leave the default settings as they are and click **Finish**.
5. In the Connection Wizard dialog that appears, select *Microsoft Access (ADO)*, and click **Next**.
6. In the next screen, select *Use an existing database*, browse for the `MyCars.mdb` DB, and click **Connect**.
7. The connection will be made, and a dialog appears in which you can select the DB table you want to use (*screenshot below*).



Add Page Source

select a Table or View

Catalog: MyCars

Cars
Cars_Manufacturer_Countries
Cars_Survey
MSysAccessObjects
MSysAccessXML
MSysACEs
MSysNameMap
MSysNavPaneGroupCategories
MSysNavPaneGroups
MSysNavPaneGroupToObjects

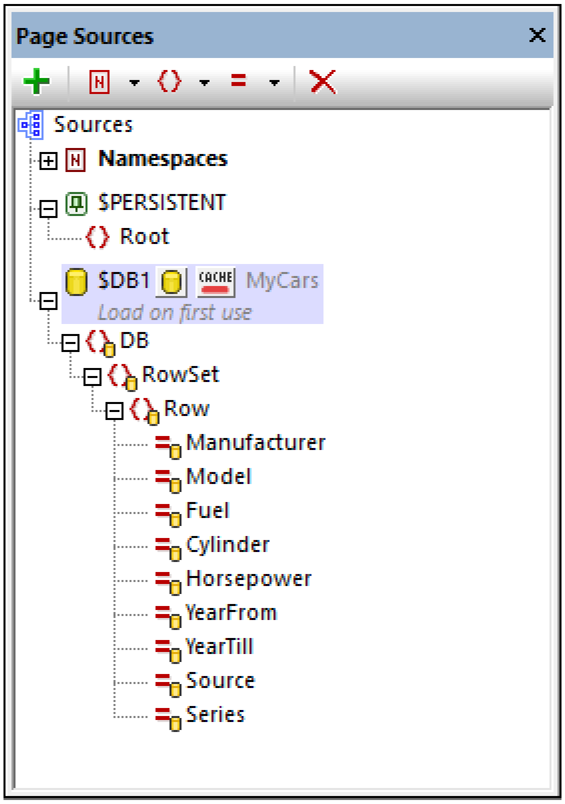
Please select one of the following options to define how the page source should be created:

1. If you wish to use all rows from chosen table/view or
   if you want to filter out specific rows =>

2. Alternatively, you can manually add any SELECT statement =>

Select the `Cars` table as shown in the screenshot, and click **Build a SELECT statement** and then **Finish**.

The `Cars` table is created as a DB page source of the design page, and is displayed in the Page Sources pane 270 (*screenshot below*).



In the next section, a combo box is created in the design so that the end user can select the manufacturer whose cars should be displayed.

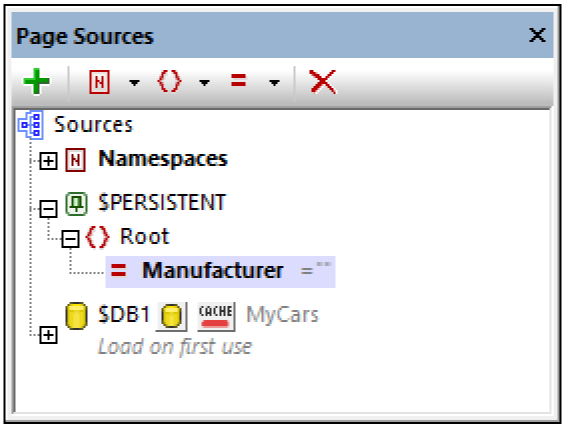
## 4.3.2 Persistent Tree to Hold User Input

To interact with the end user of the solution, a combo box[459] will be used to present the list of manufacturers. The user can select one manufacturer from among those listed in the combo box. A `$PERSISTENT` tree needs to be built so that the user selection can be stored in a node. The value in this node will later be used to make the data selection from the DB[112] to display the selected manufacturer's cars (*see next section*[112]).

This section describes how to build the `$PERSISTENT` tree and the combo box discussed above.

### Build the $PERSISTENT tree

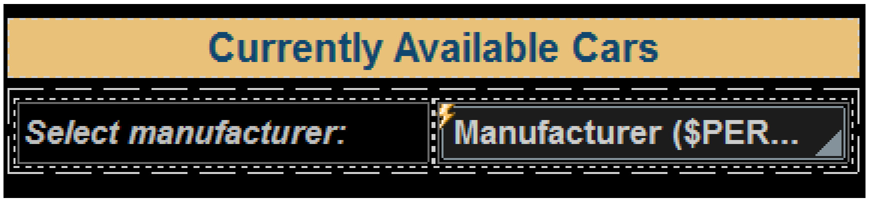
When the design is created a `$PERSISTENT` tree is automatically created in the Page Sources pane[270] with a root (or document) element called **`Root`** (*see screenshot below*). Since only a single node is required to hold the user selection, all that is needed is a child element or child attribute node. Add an attribute node called **`Manufacturer`** to the **`Root`** element as follows. Select **`Root`** in the `$PERSISTENT` tree, then click the **Add Attribute** icon in the pane's toolbar and select **Add Child Attribute**. Rename the attribute to **`Manufacturer`** (*see screenshot below*).



Notice that the attribute **`Manufacturer`** has the empty string (`""`) as its starting value (*see screenshot*). This ensures that when the `$PERSISTENT` tree is reset, the **`Manufacturer`** attribute will have a value that is the empty string.

## Create the combo box for user selection

Create the design of the page to look something like this:



The design at this point should contain:

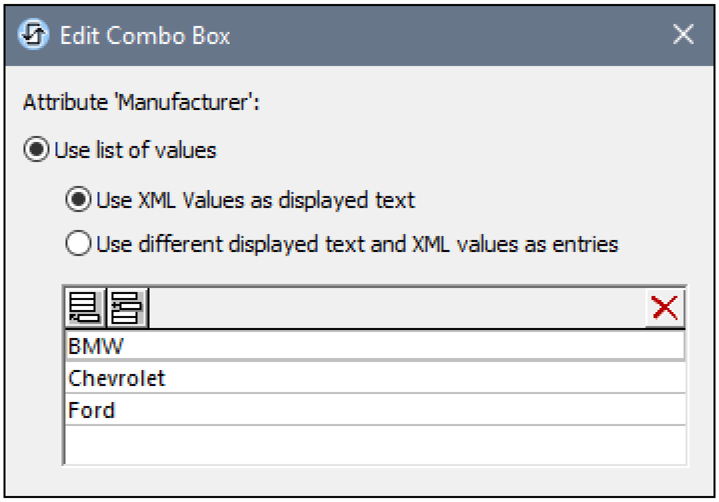
- A label 554 control bearing the title of the page.
- A static table 1063 control containing two cells.
- The left-hand table cell contains a label 554 with the words, *Select manufacturer*.
- The right-hand table cell contains the combo box 459 that will present the end user with a list of manufacturers to choose from.

To add the label 554, table 614, and combo box 459 controls, drag the respective controls 403 from the Controls Pane 266. For details about each control's properties, see the description of the respective control.

*Combo box settings*

There are three important combo box settings:

- The values that the combo box will display and the value that will be saved as the user selection. To define these values, double-click the combo box. In the Edit Combo Box dialog that appears (*screenshot below*), add three values as shown in the screenshot. These values will be the dropdown-list items of the combo box *as well as* their corresponding XML values.



Edit Combo Box

Attribute 'Manufacturer':

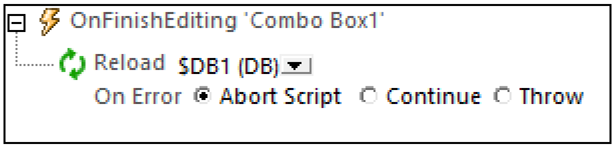
● Use list of values

● Use XML Values as displayed text

○ Use different displayed text and XML values as entries

| |
|---|
| BMW |
| Chevrolet |
| Ford |

- A page source link to store the combo box selection. Drag the `$PERSISTENT/Root/Manufacturer` attribute from the Page Sources pane[270] onto the combo box. This establishes the page source link. As a result, when the end user selects a dropdown-list item in the combo box, then the selected item's corresponding XML value will be saved to the `$PERSISTENT/Root/Manufacturer` node.
- After the end user selects a new dropdown-list item in the combo box, the `$DB1` tree should be reloaded. (This is because the data to load from the DB data source must be selected on the basis of the combo box selection. So when the combo box value changes the data selection must also change. How the DB data is selected on the basis of the combo box selection will become clearer at the end of the next section[112].) To reload the `$DB1` tree, add a Reload action[801] to the `OnFinishEditing`[665] event of the combo box[459]. Do this by right-clicking the combo box and selecting **Control Actions for OnFinishEditing**. In the Actions dialog that appears, add the Reload action[801] as shown in the screenshot below.



OnFinishEditing 'Combo Box1'

Reload $DB1 (DB)

On Error ● Abort Script ○ Continue ○ Throw

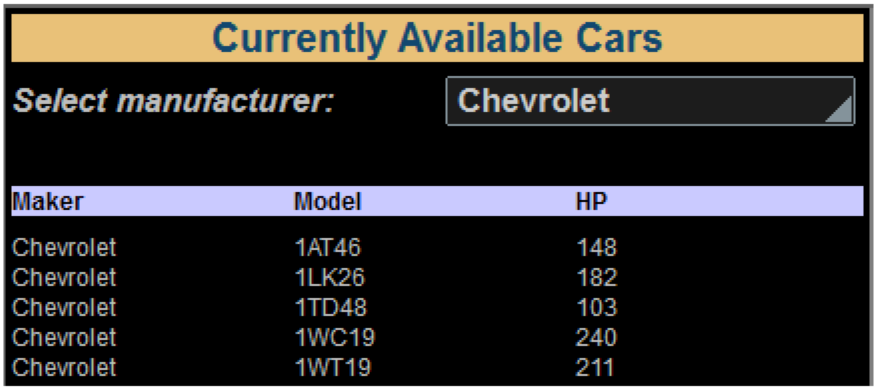
Next, we will load the selected manufacturer's cars from the DB into the solution, and display these cars in a table.

## 4.3.3 Load DB Data Based on User Selection

The DB records of car models will be displayed in a table, with only those models being loaded and displayed that belong to the manufacturer chosen by the end user (in the combo box). See the previous section[112] for details of how this is set up. In this section, first, a table is created to display the records, then the selection of records to load into the `$DB1` tree and display is defined.

### Table for displaying records

In the DB, each record (or row) corresponds to a different car model. So the best approach for displaying these records would be to add a table with repeating rows to the design, where each table row corresponds to a DB row. The table should have three columns, one each for the car manufacturer, the model, and the horsepower of the car, and also a header row (*see screenshot below*). When the user changes the selection in the combo box, then car models of the new manufacturer should be loaded and displayed.

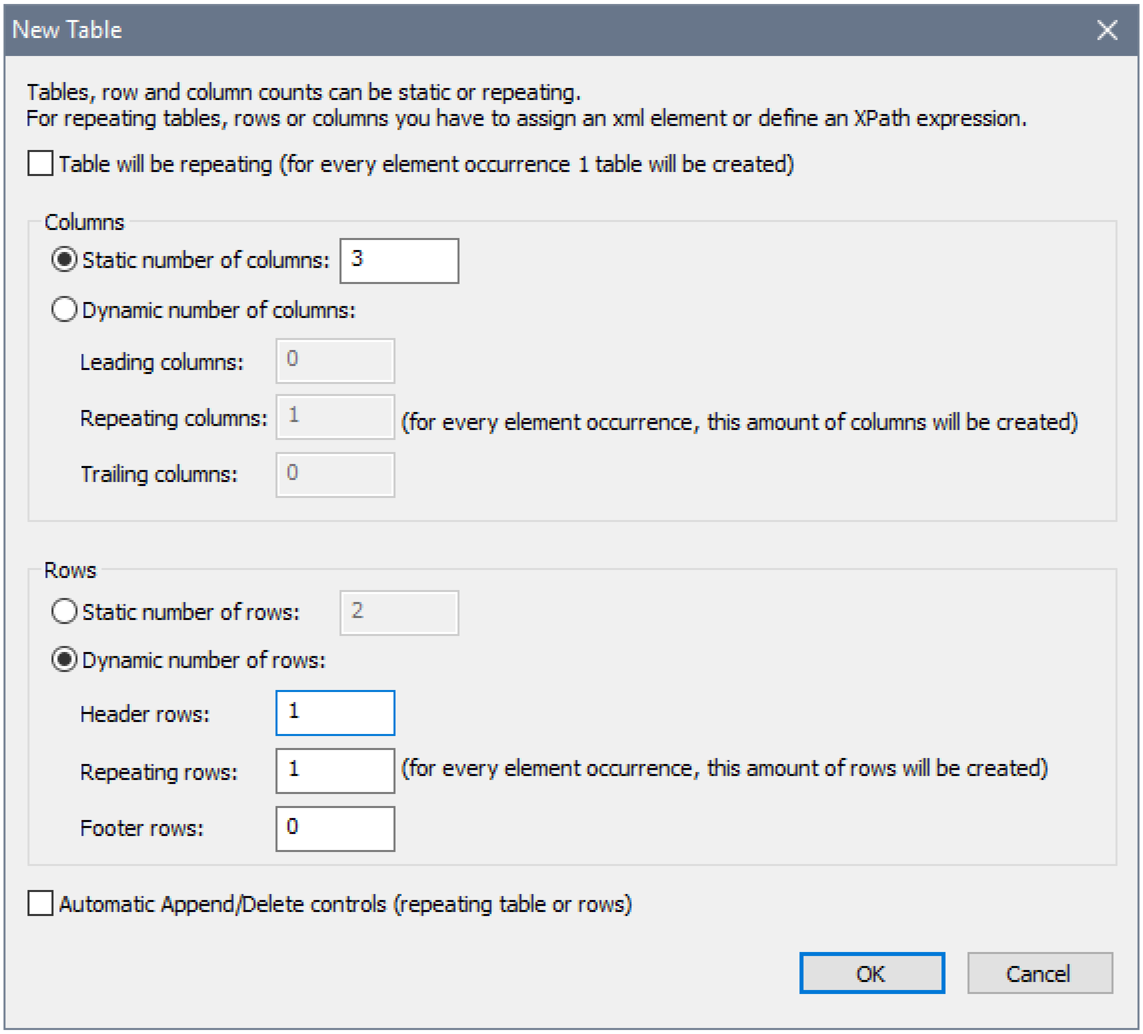


Currently Available Cars

*Select manufacturer:* Chevrolet

| Maker | Model | HP |
|---|---|---|
| Chevrolet | 1AT46 | 148 |
| Chevrolet | 1LK26 | 182 |
| Chevrolet | 1TD48 | 103 |
| Chevrolet | 1WC19 | 240 |
| Chevrolet | 1WT19 | 211 |

To insert a table that has the properties described above, add a table control with the same specifications as those shown in the screenshot below.



Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

□ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 3

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

○ Static number of rows: 2

● Dynamic number of rows:

Header rows: 1

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

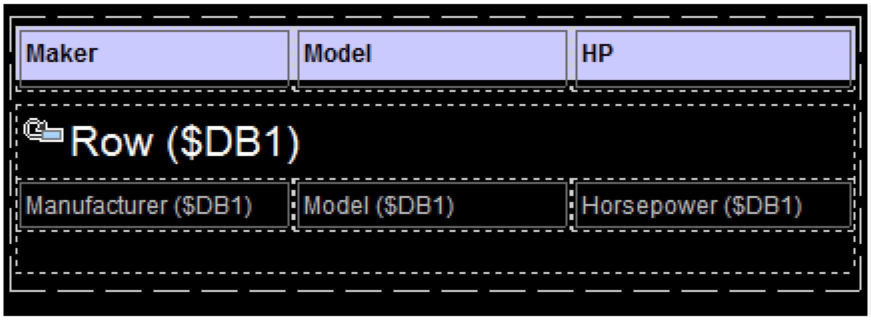
Footer rows: 0

□ Automatic Append/Delete controls (repeating table or rows)

Inside the table, do the following:

- Key the repeating row of the table to the DB row. Do this by dragging the element `$DB1/DB/RowSet/Row` onto the repeating row icon of the table. This specifies that for each (record) row in the `$DB1` tree, there will be a corresponding row in the table.
- Drag-and-drop label[554] controls into each of the three header cells and give them suitable text, corresponding to the three column headers (*see screenshot below*).
- For the contents of the three columns, drag and drop, respectively, the following attribute nodes of the `Row` element from the Page Sources pane[270], and create them as label[554] controls: **`Manufacturer`**, **`Model`**, **`Horsepower`** (*see screenshot below*).

When you have finished, the table in the design should look something like this:



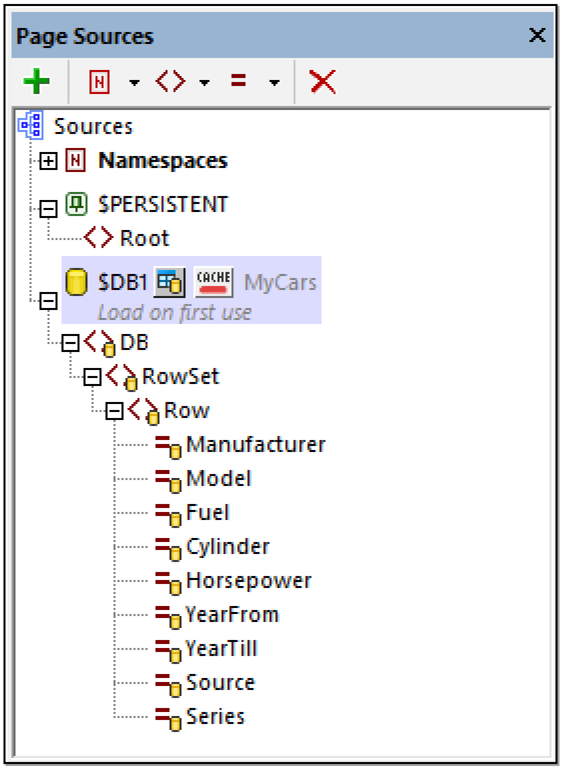
| Maker | Model | HP |
|---|---|---|
| Manufacturer ($DB1) | Model ($DB1) | Horsepower ($DB1) |

## Select the DB records to load and display

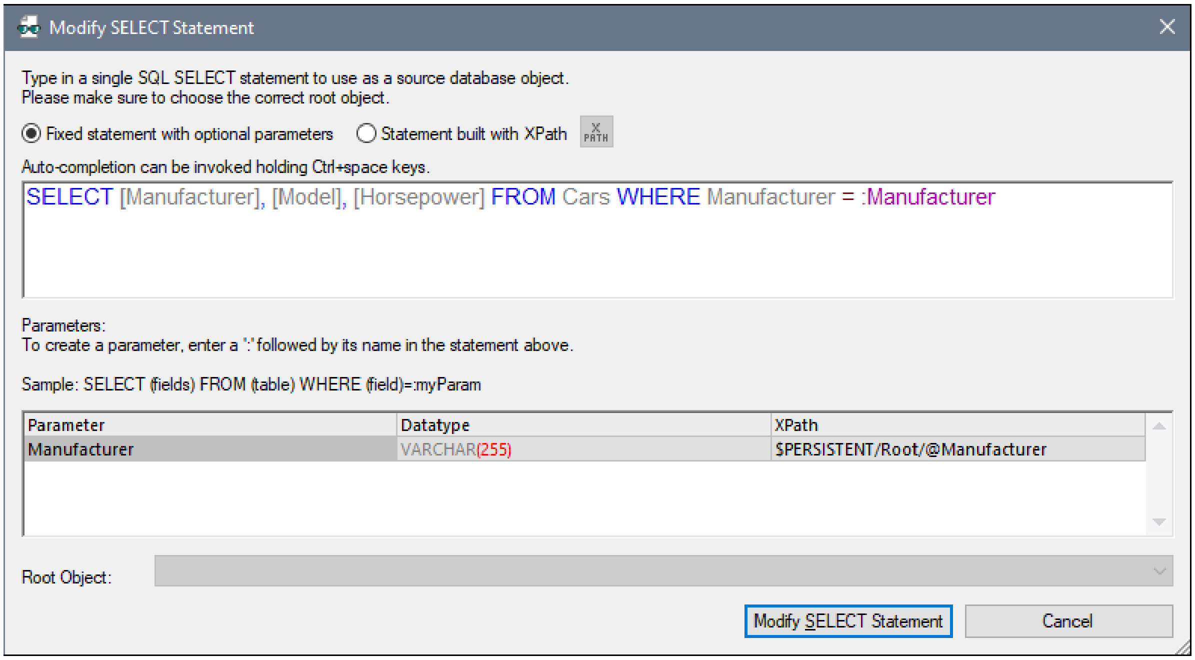
The table that you have just created will display all the records that are loaded from the DB. And the way the `$DB1` tree is currently defined, **all the records in the DB** will be loaded—that is, all the car models of **all the manufacturers**—and all will be displayed. However, the goal is to **load** and display only the car models of the manufacturer that the user selects in the combo box.

To load only the car models of the selected manufacturer, create a **`SELECT`** statement on the `$DB1` tree. Do this as follows:

1. Click the **DB** icon to the right of the **`$DB1`** tree legend (*see screenshot below*).



2. In the dialog that appears, click **Change to any SELECT**.
3. In the Modify SQL SELECT Statement dialog that appears, enter the following **SELECT** statement: `SELECT [Manufacturer], [Model], [Horsepower] FROM Cars WHERE Manufacturer = :Manufacturer`. This statement selects—and therefore loads—only the *Manufacturer*, *Model*, and *Horsepower* fields of those records where the *Manufacturer* field matches the value supplied by the **:Manufacturer** parameter. Since the **SELECT** statement contains a parameter (**:Manufacturer**), a line for the parameter definition will automatically be added to the lower pane of the dialog (*see screenshot*).



Type in a single SQL SELECT statement to use as a source database object.
Please make sure to choose the correct root object.

● Fixed statement with optional parameters ○ Statement built with XPath

Auto-completion can be invoked holding Ctrl+space keys.

```
SELECT [Manufacturer], [Model], [Horsepower] FROM Cars WHERE Manufacturer = :Manufacturer
```

Parameters:
To create a parameter, enter a ':' followed by its name in the statement above.

Sample: SELECT (fields) FROM (table) WHERE (field)=:myParam

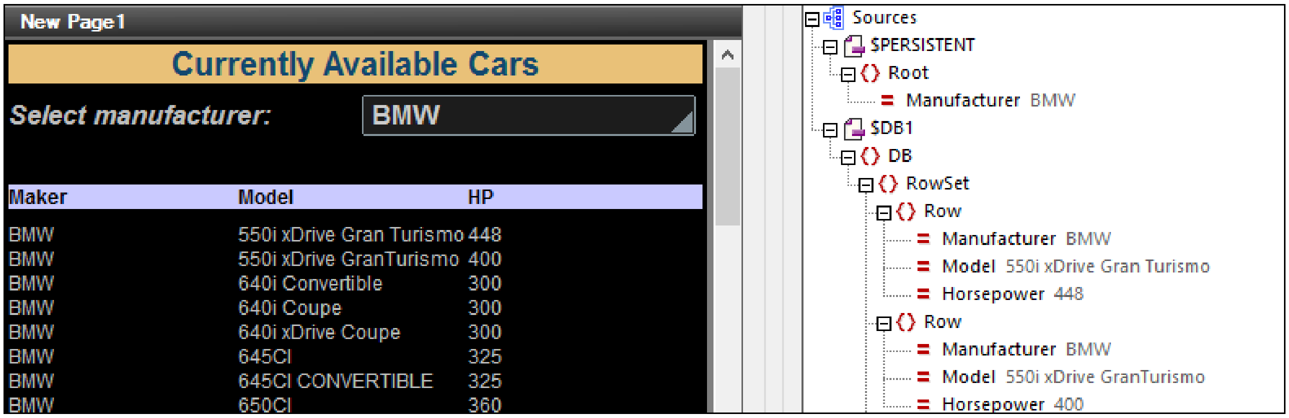
| Parameter | Datatype | XPath |
|---|---|---|
| Manufacturer | VARCHAR(255) | $PERSISTENT/Root/@Manufacturer |

Root Object:

4. Enter the following XPath expression as the definition of the **`:Manufacturer`** parameter value: `$PERSISTENT/Root/@Manufacturer`. This sets the SQL SELECT statement to select those DB records where the *Manufacturer* field matches the value currently in the `$PERSISTENT/Root/@Manufacturer` node—which is the user selection.

It is important to note that the **`SELECT`** statement that is defined on the `$DB1` page source selects what data from the DB to **load** into the `$DB1` tree. This is how the mechanism works:

- As soon as the user changes the value in the combo box, the Reload action[801] of the control's `OnFinishEditing` event reloads the `$DB1` tree (see the definition of the combo box[110]).
- The `$DB1` tree is loaded on the basis of a **`SELECT`** statement.
- This **`SELECT`** statement uses a parameter that has as its value the value of the `$PERSISTENT/Root/@Manufacturer` attribute, which holds the new user selection. The parameter causes only those DB rows to be selected that have a *Manufacturer* field value that is the same as the manufacturer which the user selected.
- All the DB rows that have been loaded into the `$DB1` tree will be displayed in the table. But since only the rows corresponding to the user selection have been loaded into the `$DB1` tree in the first place (*see screenshot below*), the table will display only the car models corresponding to the user selection.



Currently Available Cars

Select manufacturer: BMW

| Maker | Model | HP |
|---|---|---|
| BMW | 550i xDrive Gran Turismo | 448 |
| BMW | 550i xDrive GranTurismo | 400 |
| BMW | 640i Convertible | 300 |
| BMW | 640i Coupe | 300 |
| BMW | 640i xDrive Coupe | 300 |
| BMW | 645CI | 325 |
| BMW | 645CI CONVERTIBLE | 325 |
| BMW | 650CI | 360 |

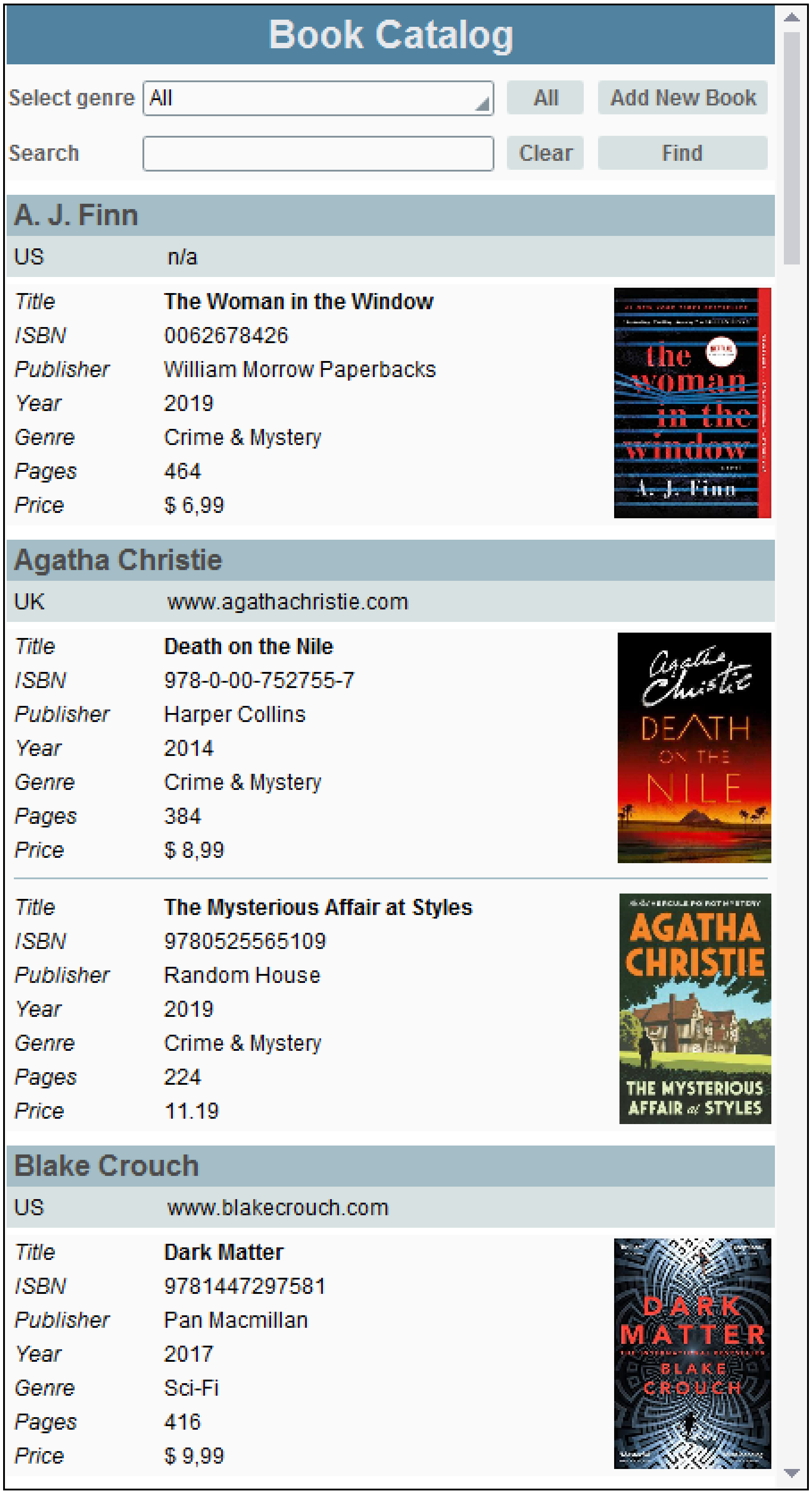
```
Sources
  $PERSISTENT
    Root
      Manufacturer  BMW
  $DB1
    DB
      RowSet
        Row
          Manufacturer  BMW
          Model  550i xDrive Gran Turismo
          Horsepower  448
        Row
          Manufacturer  BMW
          Model  550i xDrive GranTurismo
          Horsepower  400
```

# 4.4 Hierarchical Database

This tutorial describes key features of a design based on a hierarchical DB. The SQLite DB we use is a catalog of books that has two tables: *Authors* and *Books*. In the database, the Books table has a foreign key that relates it to an Author.

We would like our design to do the following (*see screenshot below*):

- Display each author in the DB, together with information about all the author's books. For example, you will see in the screenshot that Agatha Christie is listed with information about her (from the *Authors* table) and information about her two books in the catalog (from the *Books* table).
- Filter the catalog according to genre
- Edit book and author information
- Add new books to the DB
- Search the DB and filter the catalog to show books that contain the search string in any of its fields



Select genre: All

Search:

**A. J. Finn**

US n/a

| | |
|---|---|
| *Title* | **The Woman in the Window** |
| *ISBN* | 0062678426 |
| *Publisher* | William Morrow Paperbacks |
| *Year* | 2019 |
| *Genre* | Crime & Mystery |
| *Pages* | 464 |
| *Price* | $ 6,99 |

**Agatha Christie**

UK www.agathachristie.com

| | |
|---|---|
| *Title* | **Death on the Nile** |
| *ISBN* | 978-0-00-752755-7 |
| *Publisher* | Harper Collins |
| *Year* | 2014 |
| *Genre* | Crime & Mystery |
| *Pages* | 384 |
| *Price* | $ 8,99 |

| | |
|---|---|
| *Title* | **The Mysterious Affair at Styles** |
| *ISBN* | 9780525565109 |
| *Publisher* | Random House |
| *Year* | 2019 |
| *Genre* | Crime & Mystery |
| *Pages* | 224 |
| *Price* | 11.19 |

**Blake Crouch**

US www.blakecrouch.com

| | |
|---|---|
| *Title* | **Dark Matter** |
| *ISBN* | 9781447297581 |
| *Publisher* | Pan Macmillan |
| *Year* | 2017 |
| *Genre* | Sci-Fi |
| *Pages* | 416 |
| *Price* | $ 9,99 |

## The tutorial files

The files for this tutorial are located in your (72 My) Documents 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\Databases`**.

- The design file: `BookCatalog.mtd`
- The SQLite database that contains the book records: `BookCatalog.sqlite`

You can open the design file in MobileTogether Designer and run simulations 1355 in MobileTogether Designer. This tutorial assumes that you know how to work with controls and actions. It does not go step-by-step through the process of building the design, but, for each design feature, discusses the strategy behind its implementation.

## Tutorial structure

This tutorial is organized into the following sections:

- Hierarchical DB Structure 120 explains the idea of hierarchical DBs and how they are used in MobileTogether
- Pages and Page Sources 122 describe the pages of the tutorial
- Main Page: Overview 126
- MainPage: Filter by Genre 132
- Main Page: Select Book to Edit 133
- Editing Page: Overview 135
- Editing Page: Edit Text and Image Data 141
- Editing Page: Save, Cancel 145
- Add New Books 147
- Search the DB 151

## Video demos of how to build a DB-based app

The Altova website provides the following video demos that show how to build a DB-based app:

- *Building a database-driven app*: How to connect to a DB and query it, how to retrieve DB records, and how to present DB data in the form of tables
- *Working with databases, Part 2*: Adds the following functionality: how to query, view, and edit DB records
- *Working with databases, Part 3*: Lets users upload images, resize images, and save the edited images; plus, more use of tables and other controls
- *Working with databases, Part 4*: Adds functionality for record creation and deletion, and for data validation
- *Working with databases, Part 5*: Shows how to filter results without calls to the backend. Also: how to save XQuery functions for increased efficiency

# 4.4.1 Hierarchical DB Structure

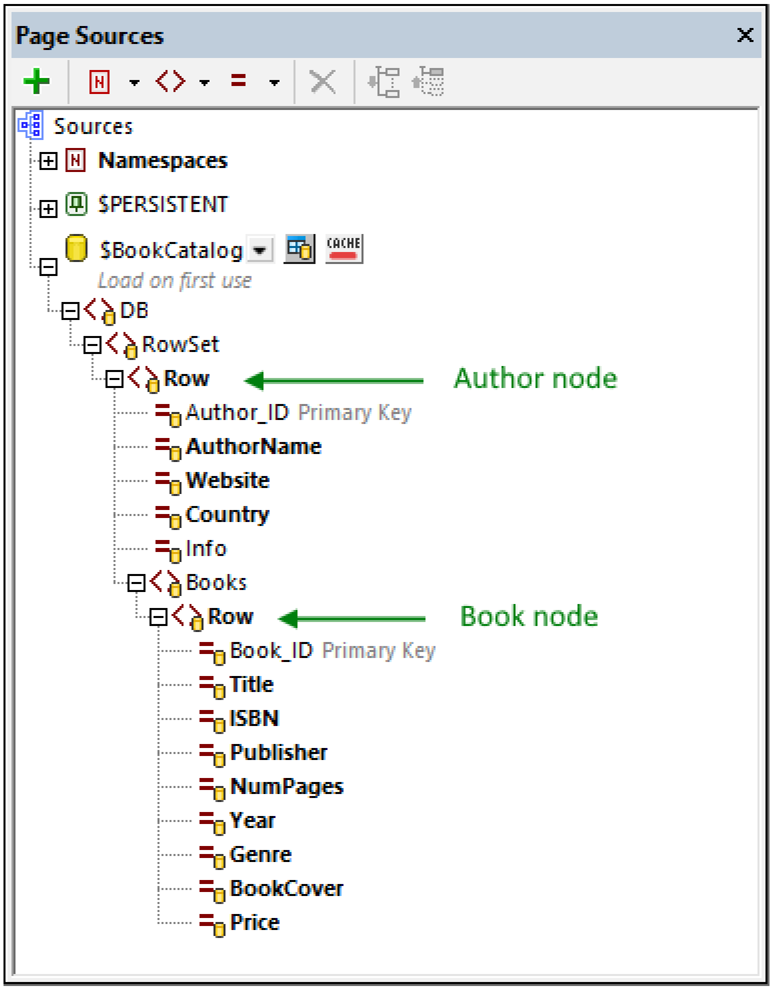
A hierarchical DB is one with a tree structure in which every node has only one parent. In MobileTogether, the *relationships* between tables are used to build the tree structure of page sources. A *relationship* between two DB tables is defined either (i) in the DB (through foreign keys) or (ii) in the design file (in the definition of the DB-page-source structure). The *relationships* can be used flexibly to define the structure of the page source. For example, an author record can have children book records. Alternatively, a book record can have children author records. The hierarchical relationship is useful when representing records  in the design. For example, an author record can be created as a table control[614] in the design, and the author's book records (from another DB table) can be created as the rows of this table control[614]. (Note the distinction between the two types of table in this discussion: *DB table* and *table control*.) Furthermore, when saving data back to the DB, the data is saved to the respective DB tables and to the correct rows of these DB tables according to the relationship between the DB tables.

In the DB for our tutorial[118], we have two tables, *Authors* and *Books*, with the following columns.

| Authors | Books |
|---|---|
| **`Author_ID (PK)`** | **`Book_ID (PK)`** |
| **`AuthorName`** | **`Title`** |
| **`Website`** | **`AuthorID (FK) = Authors.Author_ID`** |
| **`Country`** | **`ISBN`** |
| **`Info`** | **`Publisher`** |
| | **`NumPages`** |
| | **`Year`** |
| | **`Genre`** |
| | **`BookCover`** |
| | **`Price`** |

The *Books* table is related to the *Authors* table via the **`Books.AuthorID`** column, which is the *Books* table's foreign key. It links each book to an author by specifying the integer that is the appropriate **`Authors.Author_ID`** in the *Authors* table.

When we add our DB page source, we can select the *Authors* table as the main table and then specify the *Books* table as its related (child) table. This will automatically create a hierarchy because of the foreign key relationship in the DB. Each author will have zero or more book children—those book records in the DB that have a foreign key that is equal to that author's **`Author_ID`**. The hierarchical tree would be represented as in the screenshot below of our **`$BookCatalog`** page source.



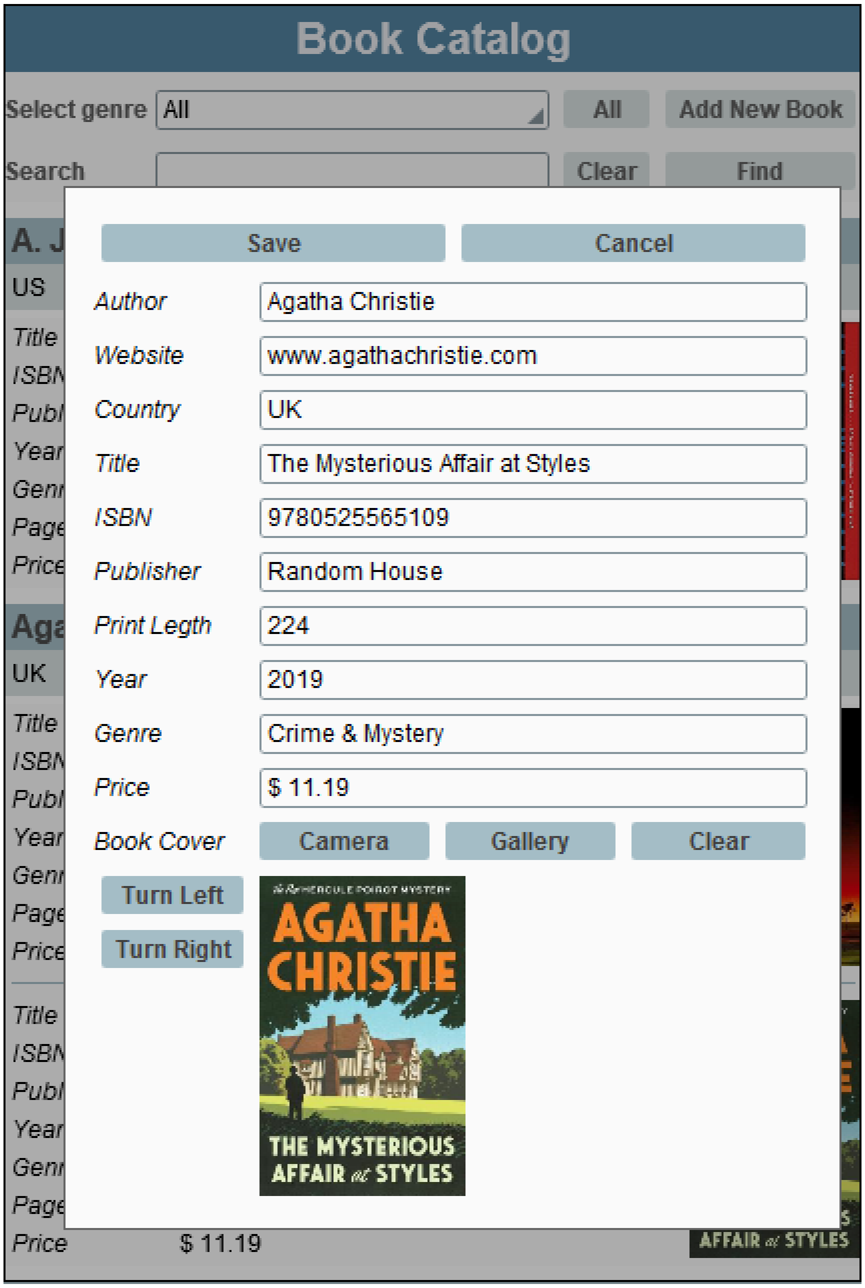
In the above tree, note that each *Author* node can have multiple *Book* child nodes—because multiple books can be related to one author (via the book's foreign key). The *Book* node in the page source represents all the books that can belong to the parent *Author* node. The *Author* node represents all the authors in the DB. In this way, the tree describes a structure of *Author* nodes with their respective *Book* children.

Also note that the foreign key of the *Books* table is automatically hidden in the page source. This is because the foreign key is the link between the tables and should not be modified.

## 4.4.2 Pages and Page Sources

The design has two pages:

- a top page named *Main Page*, which displays the book catalog (that is, the books of the DB)
- a subpage named *Editing Page*, which shows the details of the book to be edited (*see screenshot below*). This page is opened when any field of a book on the main page is clicked or when the **Add New Book** button on the main page is clicked.



*Author* Agatha Christie

*Website* www.agathachristie.com

*Country* UK

*Title* The Mysterious Affair at Styles

*ISBN* 9780525565109

*Publisher* Random House

*Print Legth* 224

*Year* 2019

*Genre* Crime & Mystery

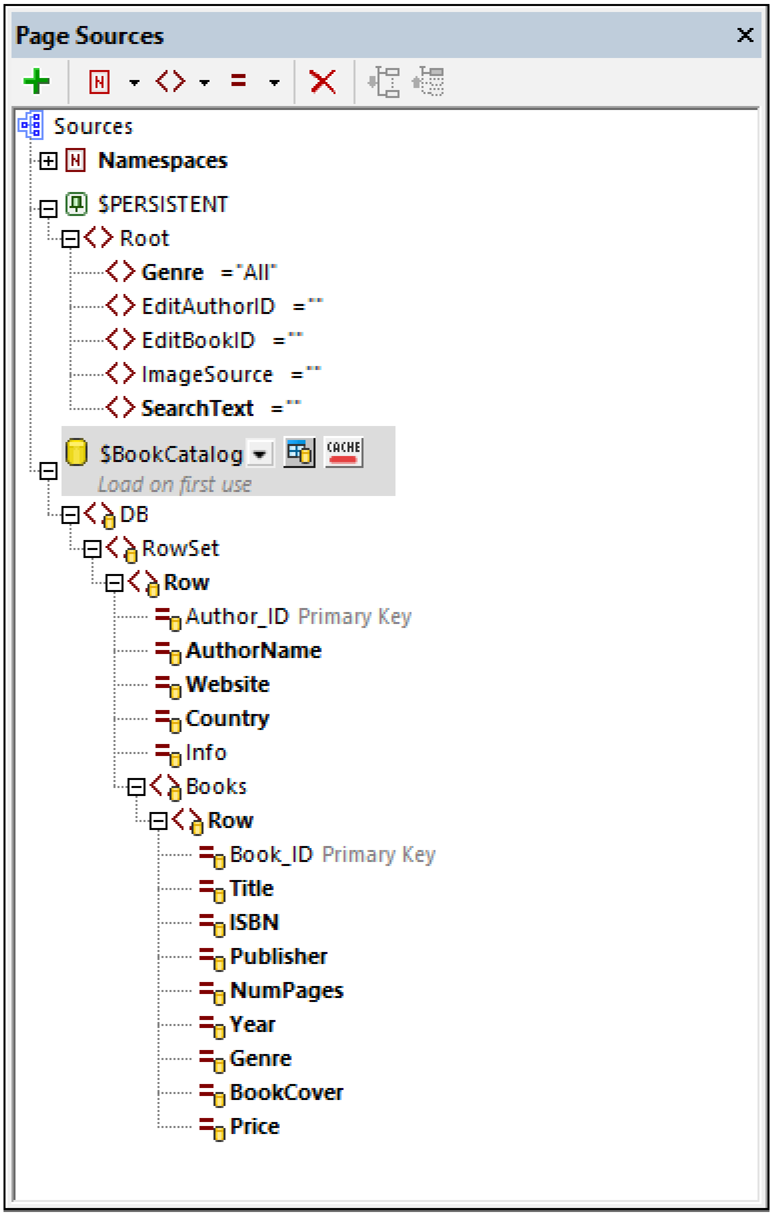
*Price* $ 11.19

*Book Cover*

## The page sources

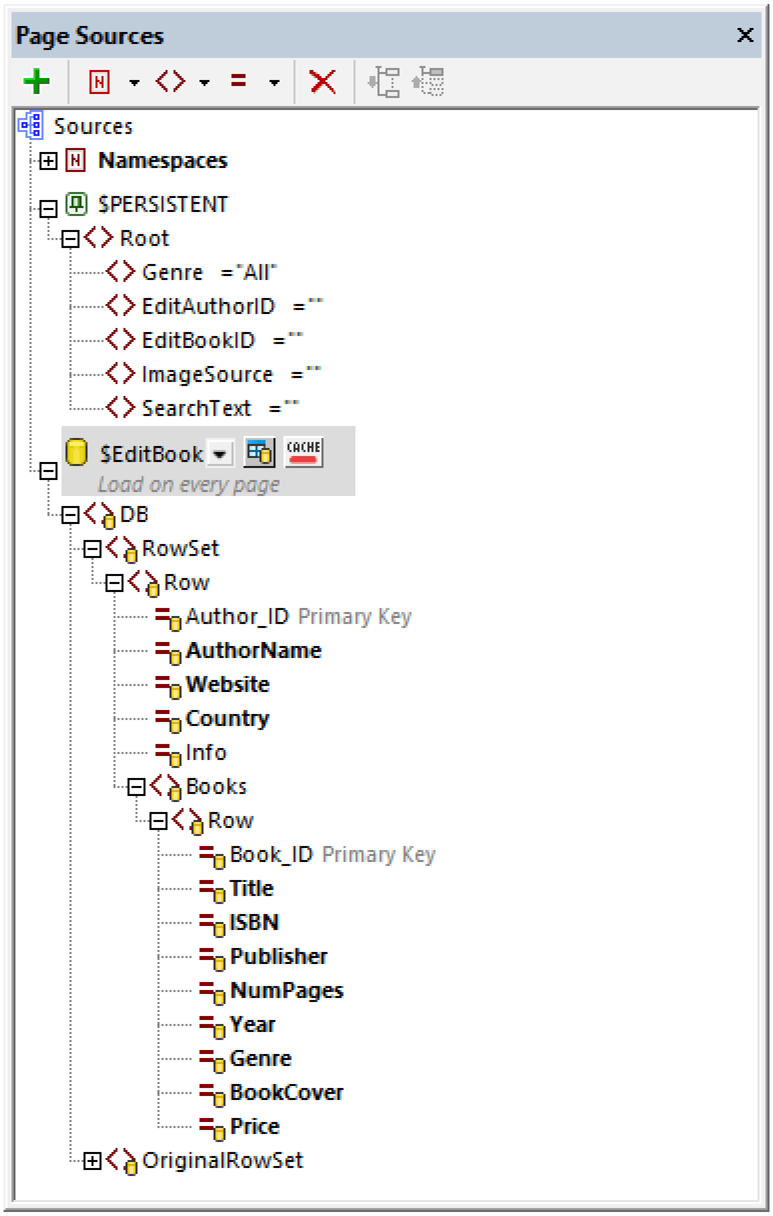
The two pages (*Main Page* and *Editing Page*), each has two page sources as shown in the screenshots below.

- *Main Page*: **`$PERSISTENT`** and **`$BookCatalog`**
- *Editing Page*: **`$PERSISTENT`** and **`$EditBook`**



Page Sources

```
Sources
  Namespaces
  $PERSISTENT
    Root
      Genre ="All"
      EditAuthorID =""
      EditBookID =""
      ImageSource =""
      SearchText =""
  $BookCatalog
  Load on first use
    DB
      RowSet
        Row
          Author_ID Primary Key
          AuthorName
          Website
          Country
          Info
          Books
            Row
              Book_ID Primary Key
              Title
              ISBN
              Publisher
              NumPages
              Year
              Genre
              BookCover
              Price
```



The key points to note are the following:

- The **`$PERSISTENT`** tree is defined once and is available on both pages. All its elements have been created with a property of *Ensure exists on load and reset* [359]. The *Genre* node has a default value of **`All`**. The reason that some nodes of this page source are bold on the main page is that these nodes have been assigned to controls on the page.
- Both **`$BookCatalog`** and **`$EditBook`** are based on the same `BookCatalog.sqlite` DB. Each page source, however, has been defined differently. While the **`$BookCatalog`** page source imports all records of the DB, the **`$EditBook`** page source of *Editing Page* will contain only the book selected for editing. See the topics Main Page: Overview [126] and Editing Page: Overview [135] to see how these page sources have been defined.

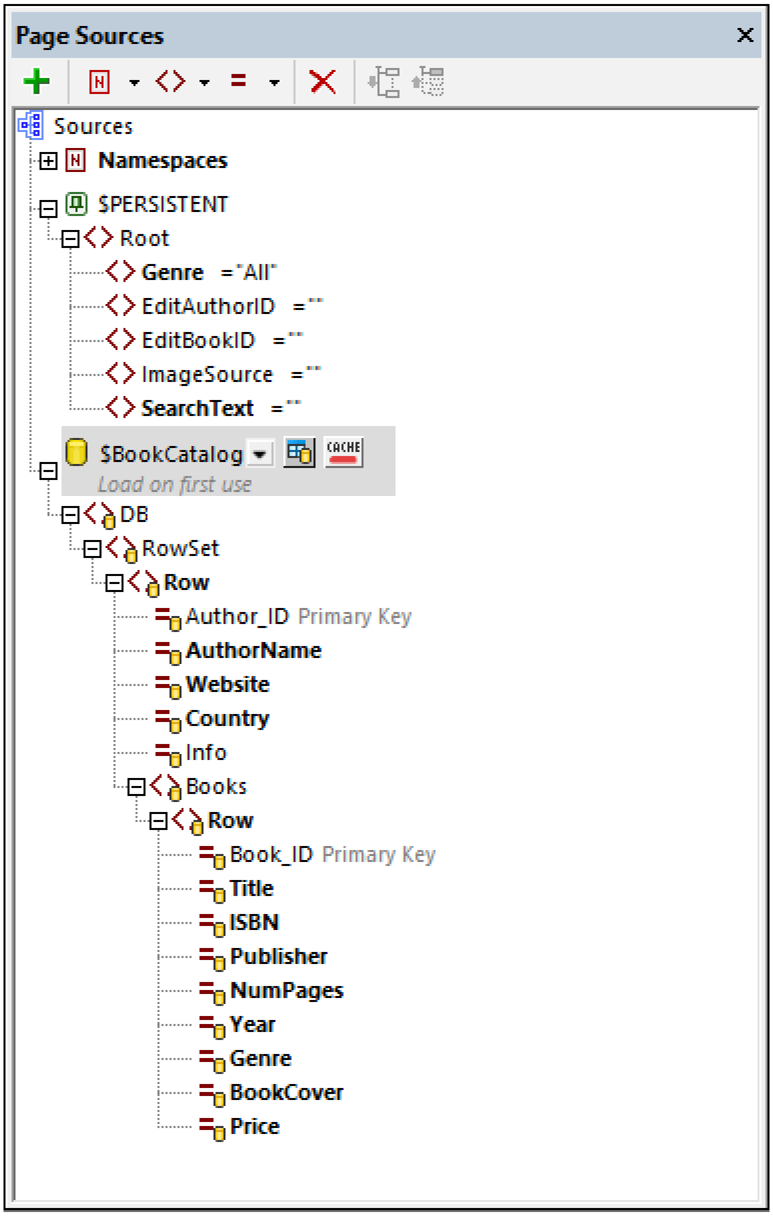
- Since the **`$EditBook`** page source of *Editing Page* will contain data that is being edited it has been created with an **`OriginalRowSet`** 359 node that will contain the data as it was before changes were made.

## 4.4.3 Main Page: Overview

The *Main Page* is the page that will be opened when the solution starts. In this topic we (i) discuss how the data for the page's sources is selected and (ii) describe the layout and function of the page design.

### Page sources

*Main Page* has two page sources: **`$PERSISTENT`** and **`$BookCatalog`** (*see screenshot below*).



*$PERSISTENT*

The **$PERSISTENT** tree is straightforward. It contains user-selected data that will be referenced to determine what actions to perform. For example: the genre selected by the user is stored in the tree's **Genre** element and the ID of the book selected for editing is stored in the **EditBookID** element.

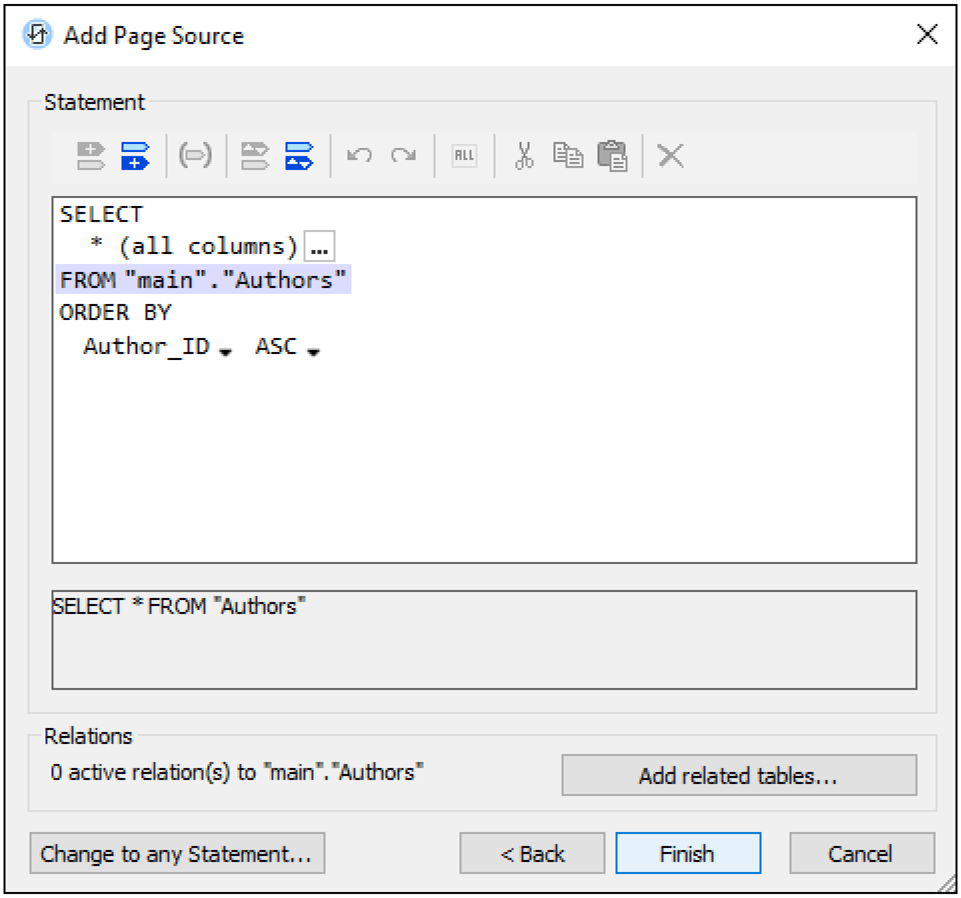
Note the following points:

- Use the toolbar of the Page Sources Pane[270] to add the new elements.
- Set the *Ensure exists on load and reset*[359] property of all elements to **true**. If you do not set this property, then the element is not automatically created in the tree and would have to be explicitly added to the tree.
- Give the *Genre* node a default value of **All** and the other elements a default value of the empty string.

*$BookCatalog*

The **`$BookCatalog`** tree will contain the data from the DB. Carry out the following steps:

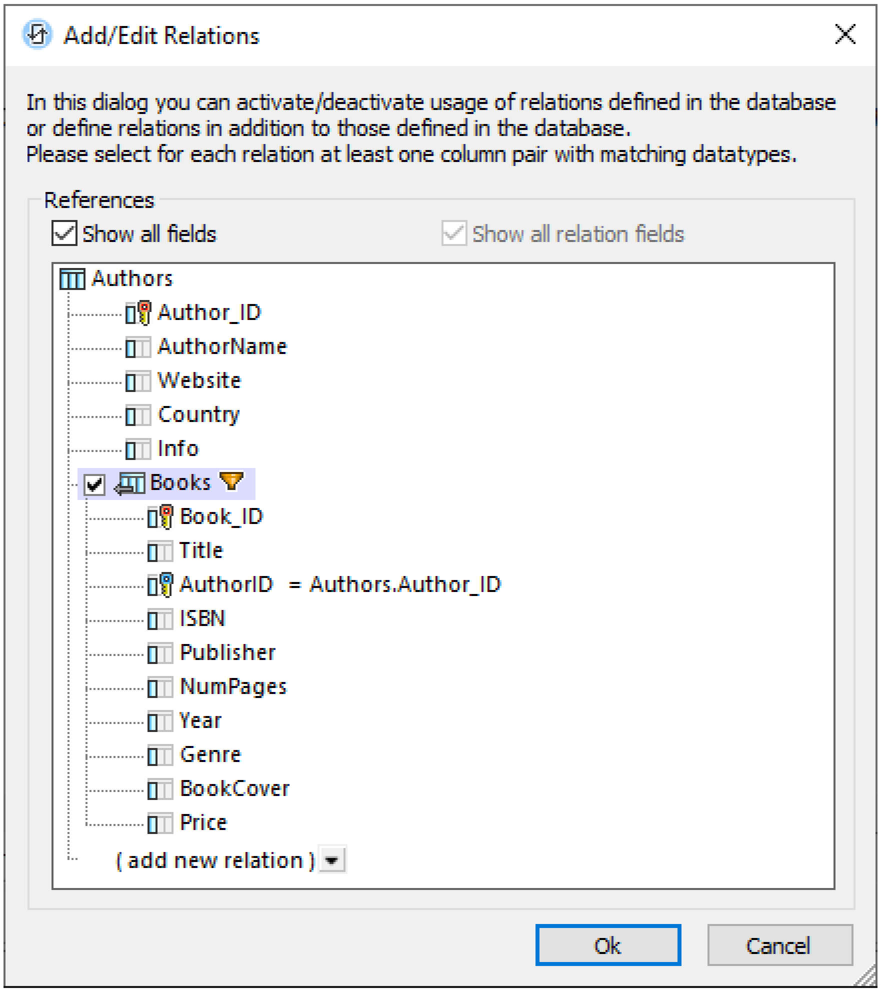
1. When you add the page source, add it as a DB structure and select the **`BookCatalog.sqlite`** database as the data source.
2. Since we want to select the author as the parent of the book, select the *Authors* table as the table to add and click **Build a SELECT Statement**.
3. In the next dialog (*screenshot below left*), append an *Order By* statement and order the Authors records by *AuthorName* and then click **Add related tables**.
4. In the Add/Edit Relations dialog that appears (*screenshot below right*), select the *Books* table, which is hierarchically linked to *Authors* via its foreign-key column *AuthorID*.
5. Make sure that the *Load Data on First Use* 359 property of the page source has been set.



```
SELECT
  * (all columns) ...
FROM "main"."Authors"
ORDER BY
  Author_ID   ASC
```

```
SELECT * FROM "Authors"
```

0 active relation(s) to "main"."Authors"



Add/Edit Relations

In this dialog you can activate/deactivate usage of relations defined in the database or define relations in addition to those defined in the database.
Please select for each relation at least one column pair with matching datatypes.

References

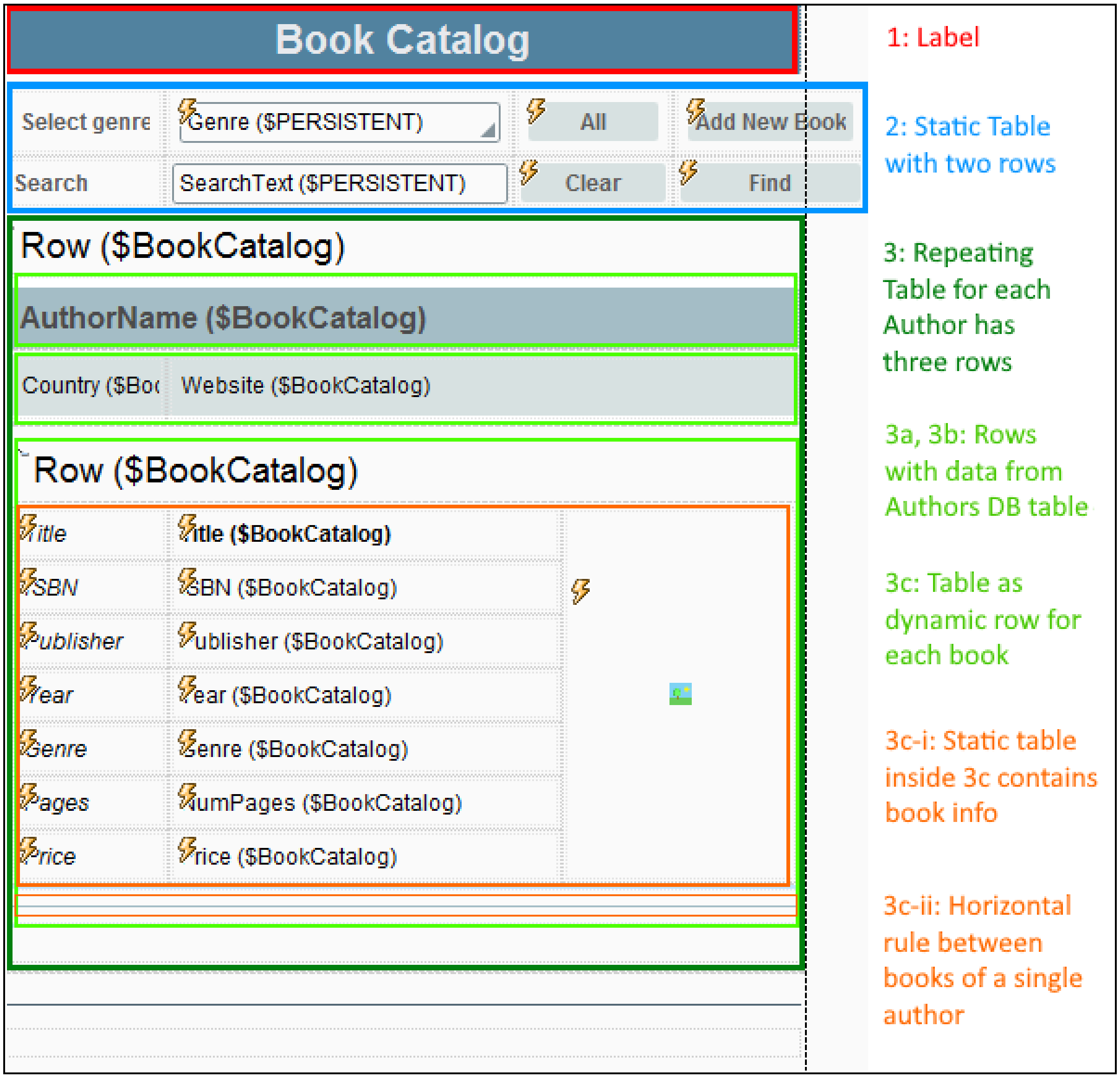
☑ Show all fields ☑ Show all relation fields

- Authors
  - Author_ID
  - AuthorName
  - Website
  - Country
  - Info
  - ☑ Books
    - Book_ID
    - Title
    - AuthorID = Authors.Author_ID
    - ISBN
    - Publisher
    - NumPages
    - Year
    - Genre
    - BookCover
    - Price
  - ( add new relation )

When the page is opened, the **`$BookCatalog`** page source will load all the authors in the DB's *Authors* tables, and each author will have child *Book* elements that are linked to the author via the book record's foreign key.

## Layout and display

The layout of the main page is shown annotated in the screenshot below. The layout consists of three main parts: (i) a label for the page title; (ii) a static table for the genre selection and search features; (iii) a repeating table for each author. At the bottom of the design is a single horizontal rule and a spacer element which serve only a presentation purpose and, therefore, will not be discussed further.



1: Label

2: Static Table with two rows

3: Repeating Table for each Author has three rows

3a, 3b: Rows with data from Authors DB table

3c: Table as dynamic row for each book

3c-i: Static table inside 3c contains book info

3c-ii: Horizontal rule between books of a single author

Note the following points:

- The Genre and Search features which are presented in the second component, the static table, are described in detail in their respective topics: Filter by Genre 132 and Search the DB 151.
- The third component is a repeating table that is associated with the *Row* element representing the Authors table. The entire table will therefore repeat for each author row (or record) in the *Authors* table of the DB.
- The repeating table for each author has three rows (light green in screenshot above).
- The first two rows display author data on Label 554 controls that are associated with pages source nodes of the *Author* element: **`AuthorName`**, **`Website`**, and **`Country`**.
- The third row of the repeating table contains a dynamic table (last bright green row in the screenshot above) that is associated with the *Book* child elements of the *Author* element. The result is that, within this third row of the repeating table for each author, the books of the author are displayed.
- Within each dynamic *Books* table, each book is displayed in its own static table (orange above).

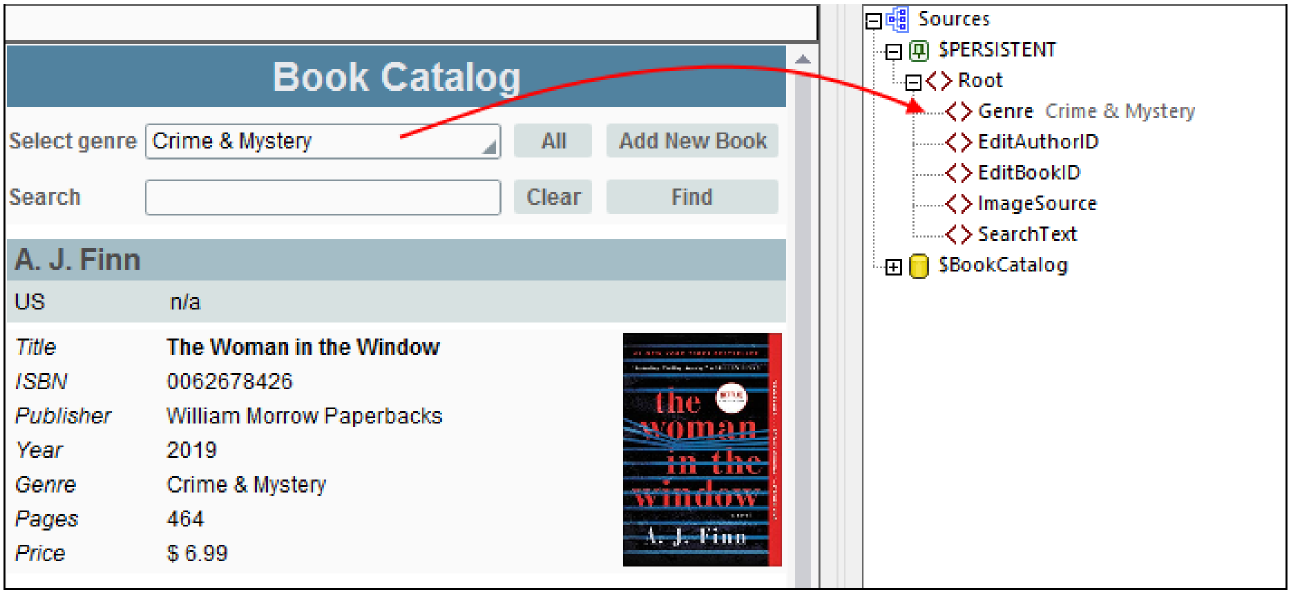
- Each book table has three columns: (i) Labels[554] bearing the name of the respective *Books* table columns; (ii) values of the respective *Books* table columns, displayed by associating Label[554] controls with the respective page source nodes of each book element; (iii) the book's cover image, which spans the rows of the static table.

## 4.4.4 MainPage: Filter by Genre

The display of the *Main Page* is filtered by genre using the mechanism given below.

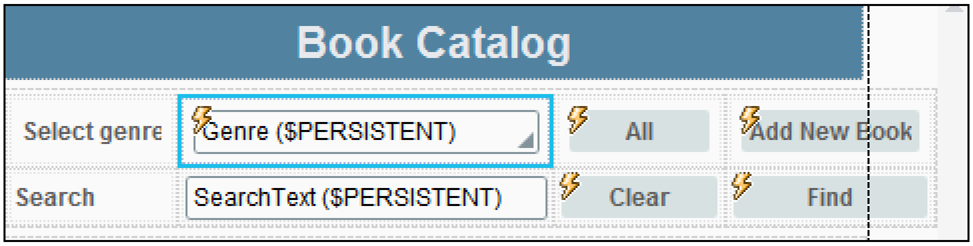
### Select a genre value

The value selected in the *Genre* combo box will be stored in the **`$Persistent/Root/Genre`** page source node.



| | |
|---|---|
| *Title* | The Woman in the Window |
| *ISBN* | 0062678426 |
| *Publisher* | William Morrow Paperbacks |
| *Year* | 2019 |
| *Genre* | Crime & Mystery |
| *Pages* | 464 |
| *Price* | $ 6.99 |

This is defined by associating the **`Genre`** page source node with the combo box (drag and drop the node onto the combo box; *see screenshot below*).



The values that are available in the dropdown list of the combo box are defined in the properties of the combo box—via an XPath expression that (i) finds the sequence of distinct values from among the list of all *Genre* values in the page source and (ii) prepends to this sequence a value of *All*: **`'All', distinct-values($BookCatalog/DB/RowSet/Row/Books/Row/@Genre)`**.

**Note:** Remember that we have set the **`$Persistent/Root/Genre`** node to have, by default, a value of *All*. See *Main Page: Overview* 126.

### Setting the visibility of authors and books by genre

In the second step of the mechanism, we first make visible in the design only those author tables that have any book with a genre that matches the selected genre. Next, since some authors may have more than one book and not all of these might belong to the selected genre, we also have to apply the Genre criterion to the visibility of each book.

Do this by selecting the corresponding design components and setting their *Visibility* property accordingly:

- For authors, we select the repeating table for Authors 126 and set the table's *Visibility* property with the following XPath expression (the context node on this component is **`Authors`**): **`if ($PERSISTENT/Root/Genre='All') then true() else Books/Row/@Genre=$PERSISTENT/Root/Genre`**. This means that when the selection is *All*, then the visibility will be true for all *Author* tables. Else, the visibility of an *Author* table will be true only if some *Book* child element of that *Author* has a genre that is equal to the currently selected genre.

- For books, we select the dynamic row for Books 126 and set this row group's *Visibility* property with the following XPath expression (the context node on this component is **`Books`**): **`if ($PERSISTENT/Root/Genre='All') then true() else @Genre=$PERSISTENT/Root/Genre`**. This means that when the selection is *All*, then the visibility will be true for all *Book* rows. Else, the visibility of a *Book* row will be true only if that book's **`@Genre`** attribute is equal to the currently selected genre.

### Resetting to All

To the right of the genre selection combo box, we have added an **All** button (*see screenshots above*), with which the display can be reset so that all the records of the DB are shown. When the button is clicked, the following actions are executed:

1. The Update Node 886 action updates the **`$Persistent/Root/Genre`** node with a value of *All*.
2. The Update Display 790 action updates the display, causing the visibility of the different design components to be recalculated (with the new *Genre* value of *All*).

To set these actions, right-click the button and select the command to access its OnClicked actions. You can run a simulation to see the effects of the button's actions.

## 4.4.5 Main Page: Select Book to Edit

When we want to edit a book, we want to bring up that book's details in an editable format. We will implement this using the following mechanism:

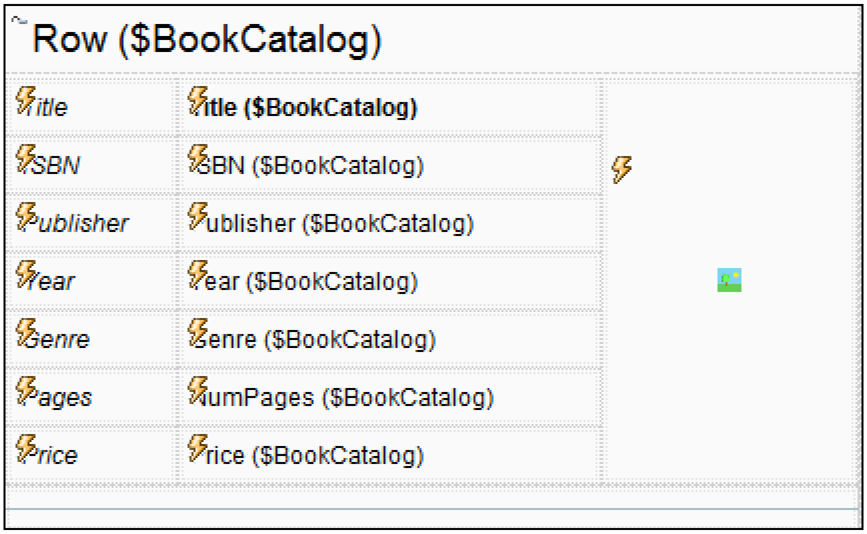
- Use a separate page, where only the details of the selected book are displayed in an editable format. In our tutorial, this is the subpage *Editing Page*.
- The DB page source of *Editing Page* must select only the particular author and the particular book to be edited. The edited data will be saved back to the respective records in the *Authors* and the *Books* tables of the DB.
- When the user selects a book to edit, we save its author ID and book ID to the **`$PERSISTENT`** tree (in the **`EditAuthorID`** and **`EditBookID`** nodes, respectively).

- These values in the **`$PERSISTENT`** tree are passed to the subpage for the selection of the data from the DB.

The steps of the mechanism are described in detail below.

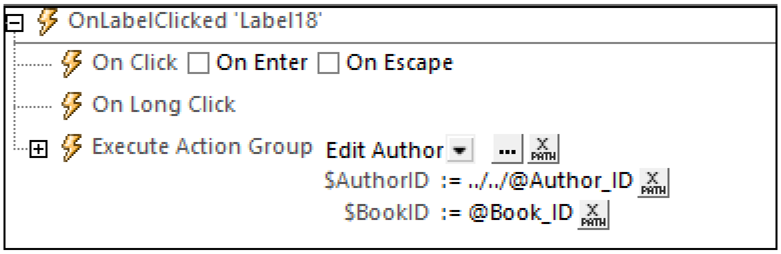
## Select IDs for book to edit

On the main page, every table cell that displays book information (*screenshot below left*) is given the same set of actions (*screenshot below right*). These are set on the control in each cell.



Row ($BookCatalog)

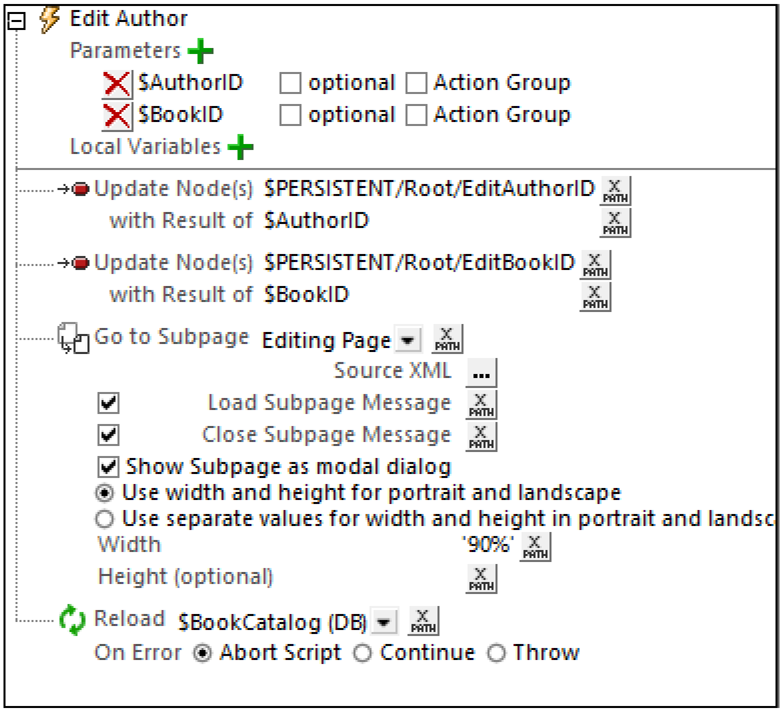
| | | |
|---|---|---|
| ⚡itle | ⚡itle ($BookCatalog) | ⚡ |
| ⚡SBN | ⚡SBN ($BookCatalog) | |
| ⚡ublisher | ⚡ublisher ($BookCatalog) | |
| ⚡ear | ⚡ear ($BookCatalog) | |
| ⚡enre | ⚡enre ($BookCatalog) | |
| ⚡ages | ⚡umPages ($BookCatalog) | |
| ⚡rice | ⚡rice ($BookCatalog) | |



```
⊟ ⚡ OnLabelClicked 'Label18'
    ⚡ On Click □ On Enter □ On Escape
    ⚡ On Long Click
  ⊞ ⚡ Execute Action Group  Edit Author ▾  ...
                             $AuthorID := ../../@Author_ID
                               $BookID := @Book_ID
```

Two parameters are passed to the Action Group *Edit Author*: **`$AuthorID`** and **`$BookID`** (*see screenshot above right*). These parameters are given values via XPath expressions that return the selected book's author ID and book ID, respectively (*see screenshot above right*).

The *Edit Author* Action Group (*screenshot below*) does the following: (i) updates the **`$PERSISTENT`** tree nodes that hold the ID information about the currently selected book; (ii) opens the *Editing Page* subpage as a modal dialog; (iii) after the edited record has been saved from the subpage and the subpage has been closed, the main page is refreshed with the reloaded **`$BoookCatalog`** page source. Consequently, the newly edited data will be displayed immediately in the main page.



Edit Author
Parameters
$AuthorID □ optional □ Action Group
$BookID □ optional □ Action Group
Local Variables

Update Node(s) $PERSISTENT/Root/EditAuthorID
with Result of $AuthorID

Update Node(s) $PERSISTENT/Root/EditBookID
with Result of $BookID

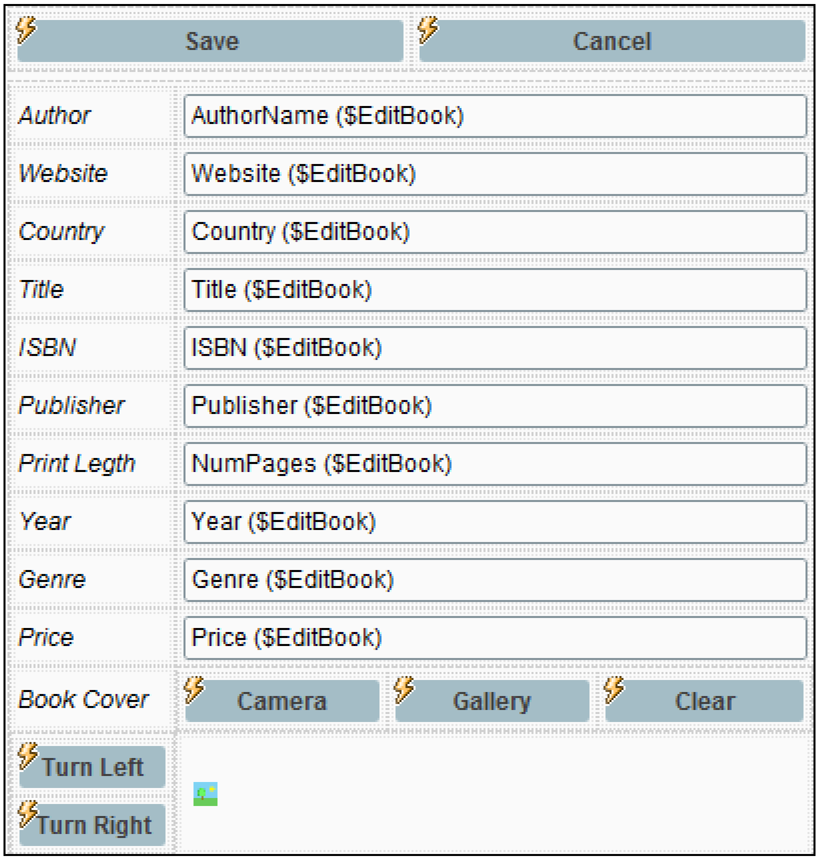
Go to Subpage Editing Page
Source XML
☑ Load Subpage Message
☑ Close Subpage Message
☑ Show Subpage as modal dialog
● Use width and height for portrait and landscape
○ Use separate values for width and height in portrait and landsc
Width '90%'
Height (optional)

Reload $BookCatalog (DB)
On Error ● Abort Script ○ Continue ○ Throw

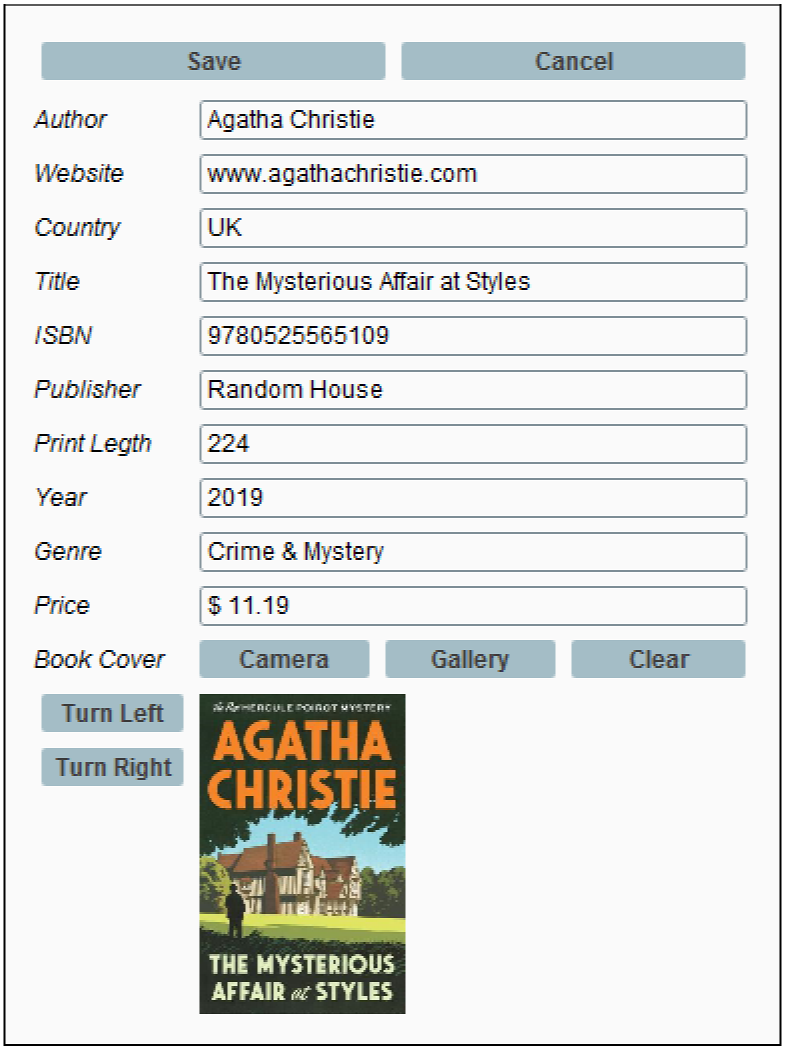
How data is selected for the subpage is described in the topic Editing Page: Overview[135].

## 4.4.6 Editing Page: Overview

The *Editing Page* will display the book to be edited (the book that was clicked on the main page). Book details can be edited and the changes saved back to the DB. The screenshots below show the page design (*left*) and the page in a simulation (*right*).



| | |
|---|---|
| *Author* | AuthorName ($EditBook) |
| *Website* | Website ($EditBook) |
| *Country* | Country ($EditBook) |
| *Title* | Title ($EditBook) |
| *ISBN* | ISBN ($EditBook) |
| *Publisher* | Publisher ($EditBook) |
| *Print Legth* | NumPages ($EditBook) |
| *Year* | Year ($EditBook) |
| *Genre* | Genre ($EditBook) |
| *Price* | Price ($EditBook) |
| *Book Cover* | |



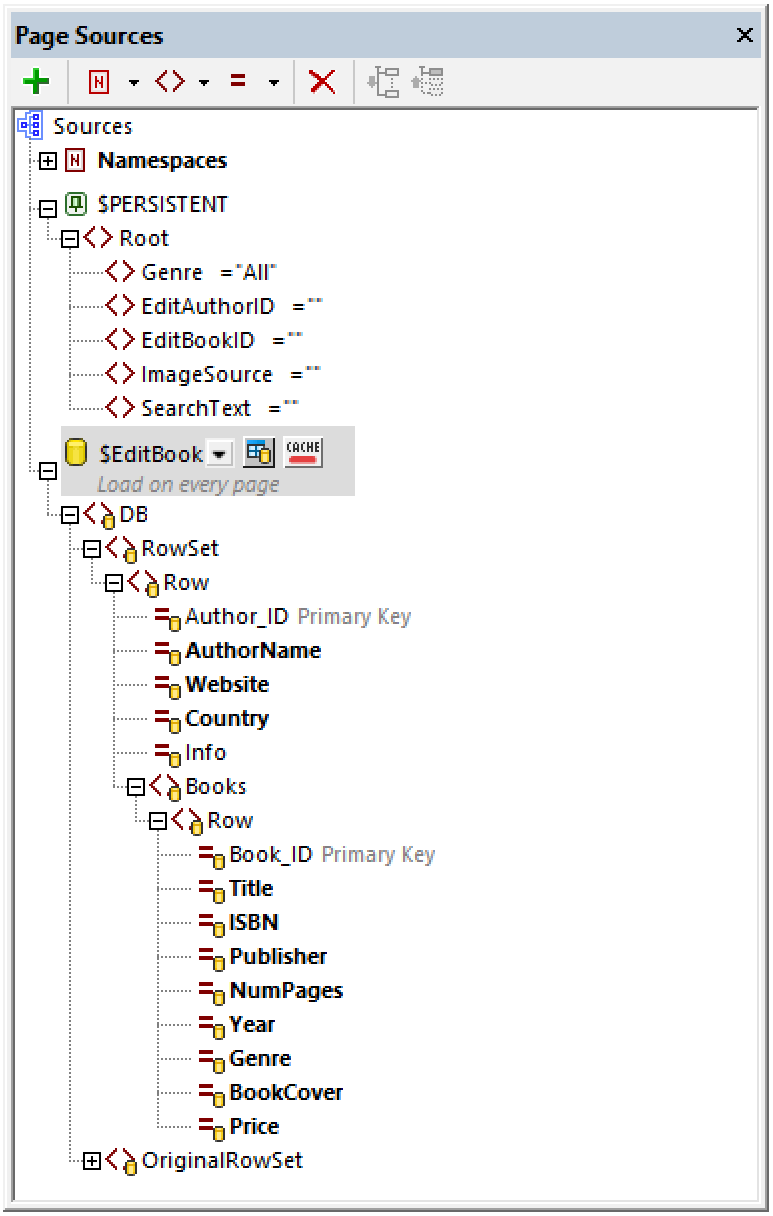
| | |
|---|---|
| *Author* | Agatha Christie |
| *Website* | www.agathachristie.com |
| *Country* | UK |
| *Title* | The Mysterious Affair at Styles |
| *ISBN* | 9780525565109 |
| *Publisher* | Random House |
| *Print Legth* | 224 |
| *Year* | 2019 |
| *Genre* | Crime & Mystery |
| *Price* | $ 11.19 |
| *Book Cover* | |

An important part of the design of this page is to restrict the display to only show the book that was selected for editing. A convenient way to do this is to configure the **`$EditBook`** page source to select only the book we want. This way, after changes have been made, saving the page source will save the edited Author and Book record to the DB.

How to configure the page source to select only a single book record is explained below.

## Page sources

*Editing Page* has two page sources: **`$PERSISTENT`** and **`$EditBook`** (*see screenshot below*).



Page Sources

```
Sources
  Namespaces
  $PERSISTENT
    Root
      Genre  ="All"
      EditAuthorID  =""
      EditBookID  =""
      ImageSource  =""
      SearchText  =""
  $EditBook
  Load on every page
    DB
      RowSet
        Row
          Author_ID Primary Key
          AuthorName
          Website
          Country
          Info
          Books
            Row
              Book_ID Primary Key
              Title
              ISBN
              Publisher
              NumPages
              Year
              Genre
              BookCover
              Price
      OriginalRowSet
```

*$PERSISTENT*

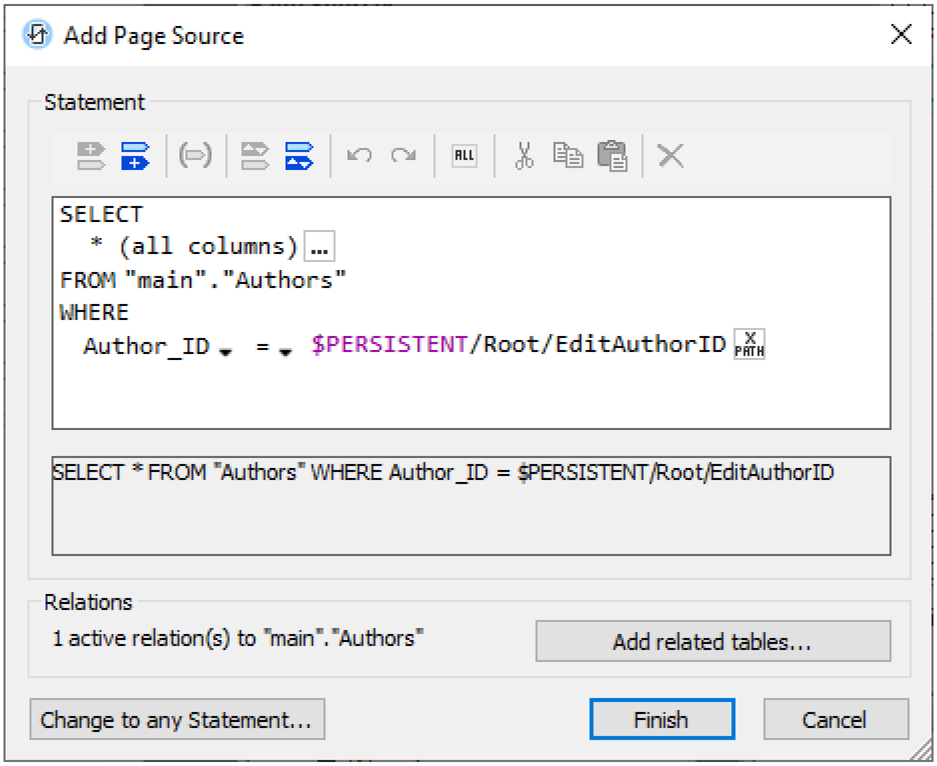
The **$PERSISTENT** tree is the same that was created for *Main Page* 126.

*$EditBook*

The **$EditBook** page source tree selects the Book record and related Author record to edit. Configure the page source as follows:

1. When you add the page source, add it as a DB structure and select the **BookCatalog.sqlite** database as the data source.
2. Since we want to select the author as the parent of the book, select the *Authors* table as the table to add and click **Build a SELECT Statement**.

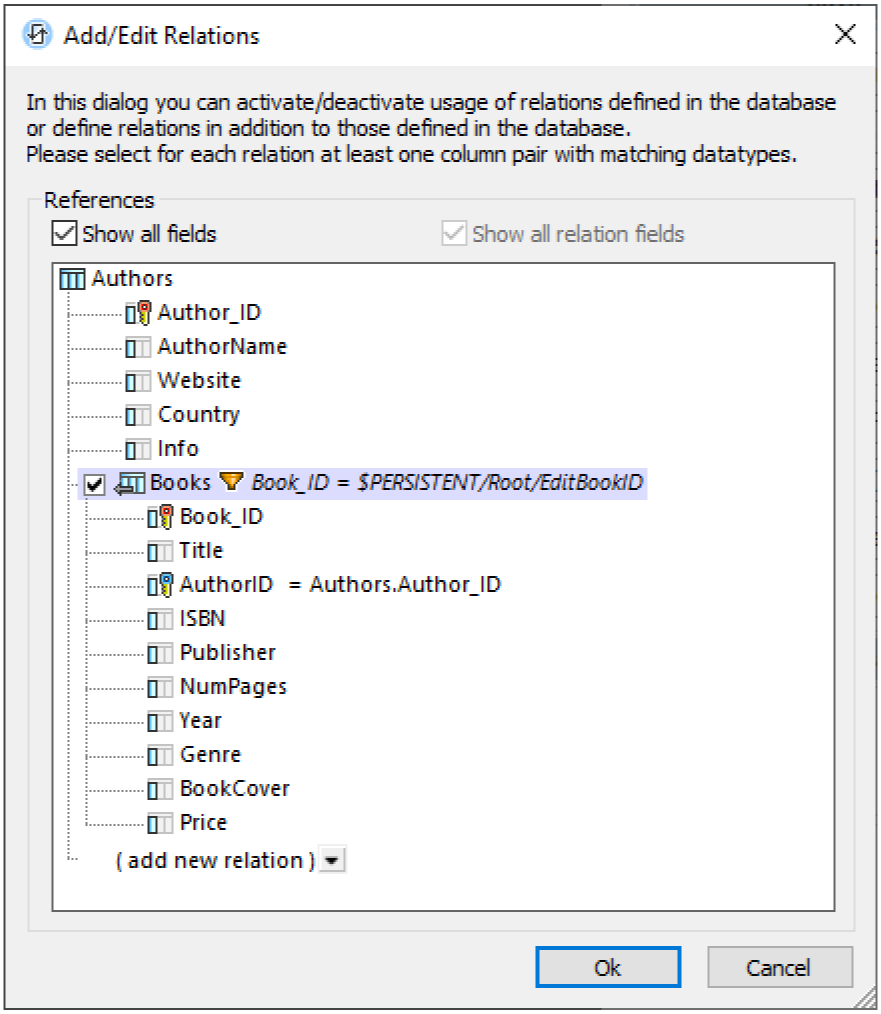
3. In the next dialog (*screenshot below left*), add a WHERE expression to select the author that has an ID which is the same as that stored in the EditAuthorID element[133] of the **`$PERSISTENT`** tree. Then click **Add related tables**.
4. In the Add/Edit Relations dialog that appears (*screenshot below right*), select the *Books* table, which is hierarchically linked to *Authors* via its foreign-key column *AuthorID*.
5. Click the table's *Filter* icon to select only that book which has a **`Book_ID`** element which is the same as that stored in the EditBookID element[133] of the **`$PERSISTENT`** tree. Without this filter, all books of the selected author will be displayed.
6. Make sure that the *Load Data on Every Page*[359] property of the page source has been set.



```
SELECT
  * (all columns)
FROM "main"."Authors"
WHERE
  Author_ID = $PERSISTENT/Root/EditAuthorID
```

SELECT * FROM "Authors" WHERE Author_ID = $PERSISTENT/Root/EditAuthorID

1 active relation(s) to "main"."Authors"



Add/Edit Relations

In this dialog you can activate/deactivate usage of relations defined in the database or define relations in addition to those defined in the database.
Please select for each relation at least one column pair with matching datatypes.

References

☑ Show all fields ☑ Show all relation fields

- Authors
  - Author_ID
  - AuthorName
  - Website
  - Country
  - Info
  - ☑ Books Book_ID = $PERSISTENT/Root/EditBookID
    - Book_ID
    - Title
    - AuthorID = Authors.Author_ID
    - ISBN
    - Publisher
    - NumPages
    - Year
    - Genre
    - BookCover
    - Price
  - ( add new relation )

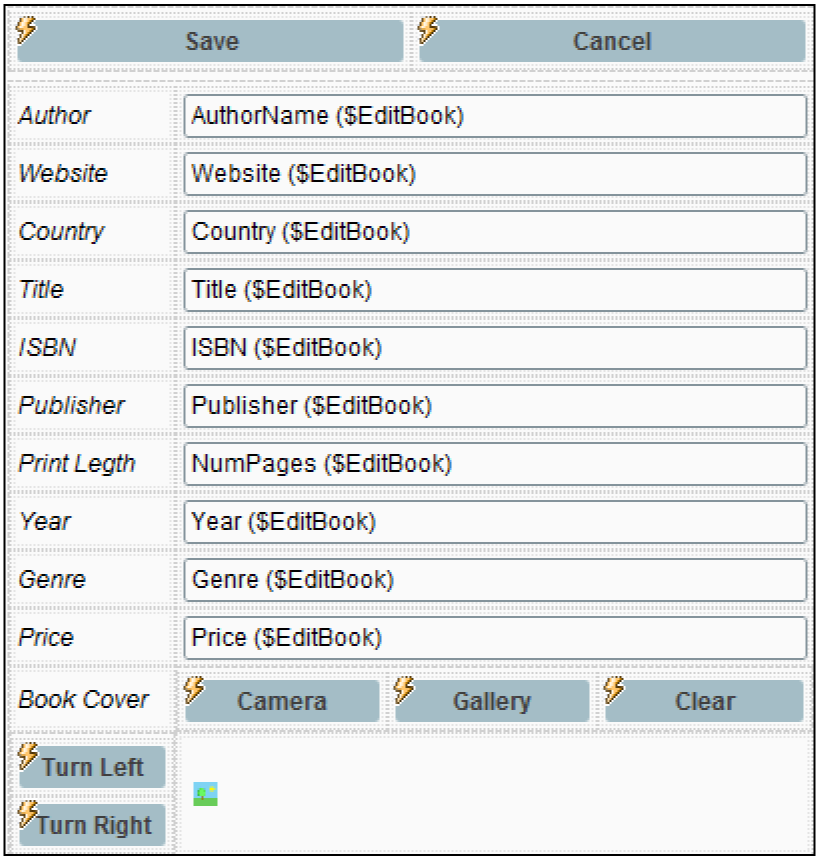
When the subpage is opened, it will display the author that has an ID matching the ID in **`$PERSISTENT/Root/EditAuthorID`** and the book that has an ID matching the ID in **`$PERSISTENT/Root/EditBookID`**. The details of this author and this book can now be displayed in *Editing Page* and edited (*see the next topic, Editing Page: Edit Text and Image Data*[141])[141].

## 4.4.7 Editing Page: Edit Text and Image Data

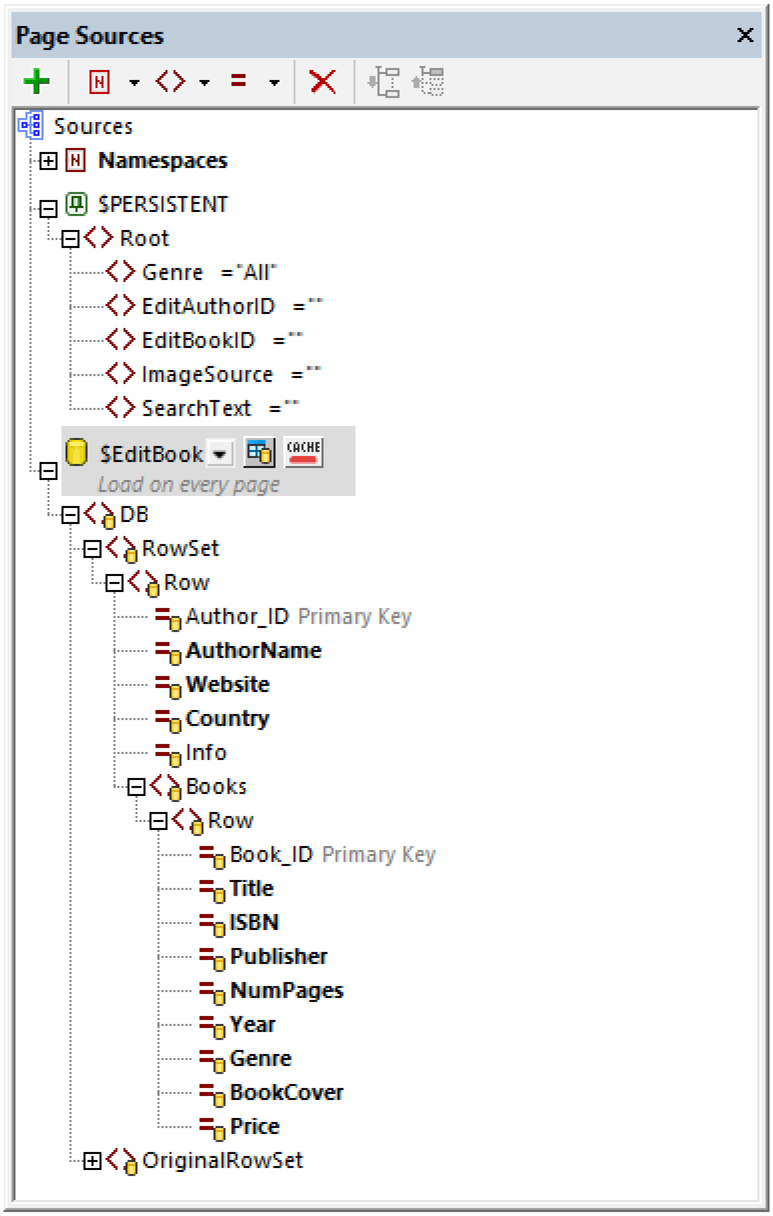
After you have configured the $EditBook page source[135] to select the record that is to be edited, create the design of the page as follows (*see screenshot below left*):

1. Add two static tables. The first table has one row with two cells: for the **Save** and **Cancel** buttons. The second table has 12 rows; these will contain the author data and book data that is to be edited.
2. In the first column of the second static table, add Label[554] controls that show the names of the data columns (*see screenshot below left*).
3. In the second column, add Edit Field[495] controls and associate them with nodes of the **`$EditBook`** page source. Associate each control–node pair by dragging-and-dropping page source nodes onto the

respective Edit Fields[495]. Page source nodes that have been associated in this way will be shown bold (see *screenshot below right*).



| | |
|---|---|
| *Author* | AuthorName ($EditBook) |
| *Website* | Website ($EditBook) |
| *Country* | Country ($EditBook) |
| *Title* | Title ($EditBook) |
| *ISBN* | ISBN ($EditBook) |
| *Publisher* | Publisher ($EditBook) |
| *Print Legth* | NumPages ($EditBook) |
| *Year* | Year ($EditBook) |
| *Genre* | Genre ($EditBook) |
| *Price* | Price ($EditBook) |
| *Book Cover* | |



## Edit text data

The edit fields for text data are all associated with page source nodes (*see above*). As a result, any text changes that the user makes will be written to the corresponding page source nodes. Notice that nodes from both the *Authors* DB table and *Books* DB table have been placed on the page.

Since the edited data is in the page source, saving the page source will update the DB. The action to save the page source has been set on the Save button 145 (*see screenshot above left*).

### Edit image data

The last two rows of the static data table are for the image (*see screenshot above left*). The first of these rows contain buttons that enable image data to be selected or cleared. The second row contains (i) buttons that enable the user to rotate the image and (ii) the Image 541 control.

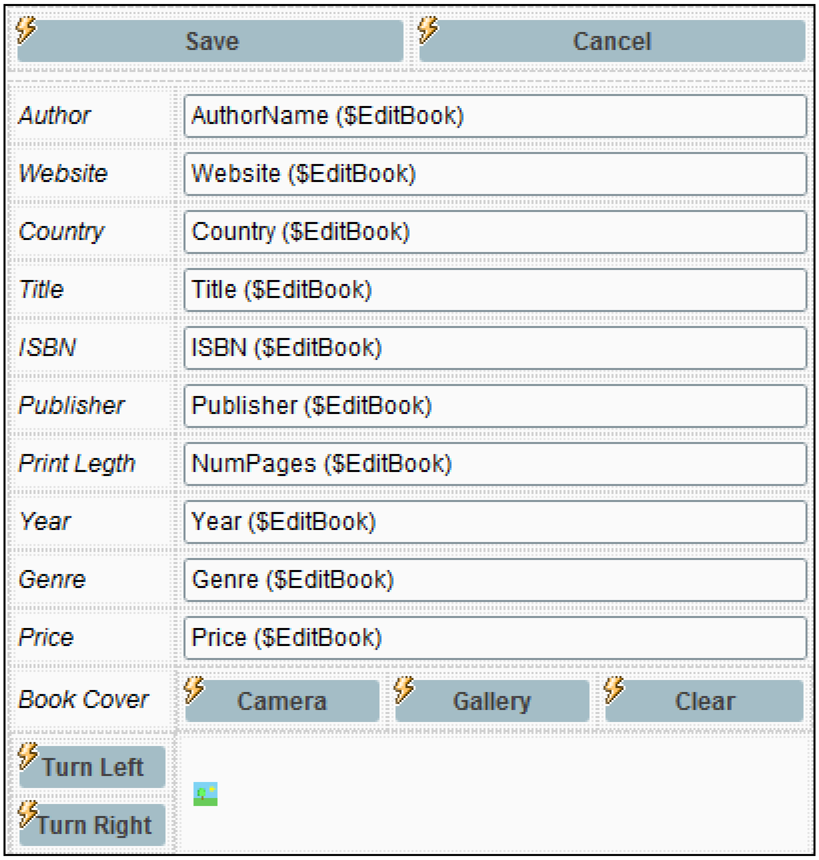
Note the following points:

- The image control is associated with the book's **`@BookCover`** node.
- The image control has its *Image Source Type* property set to `base64`.
- The **Camera** and **Gallery** buttons have the following actions set on their OnClick event: (a) Let User Choose Image 706, with the target node for the chosen image being the **`$PERSISTENT/Root/ImageSource`** node. The image will be saved here in Base64 format; (b) An Action Group that (i) deletes the **`@BookCover`** node from the **`$EditBook`** page source (effectively deleting the old image) and (ii) appends a new **`@BookCover`** node to **`$EditBook`** that contains the Base64 image data in **`$PERSISTENT/Root/ImageSource`** (effectively adding the new  image to the **`$EditBook`** page source). Open the OnButtonClicked actions of these buttons to see how they have been defined.
- The **Clear** button deletes the **`@BookCover`** node from the **`$EditBook`** page source (effectively deleting the old image).
- The **Turn Left** and **Turn Right** buttons use the MobileTogether extension function **`mt-transform-image`** 1262 to rotate the image anti-clockwise and clockwise, respectively, 90 degrees per click. Open the OnButtonClicked actions of these buttons to see how they have been defined.

After an image has been modified in the **`$EditBook`** page source, it must still be saved to the DB. This is done with the action defined on the Save button 145.

## 4.4.8 Editing Page: Save, Cancel

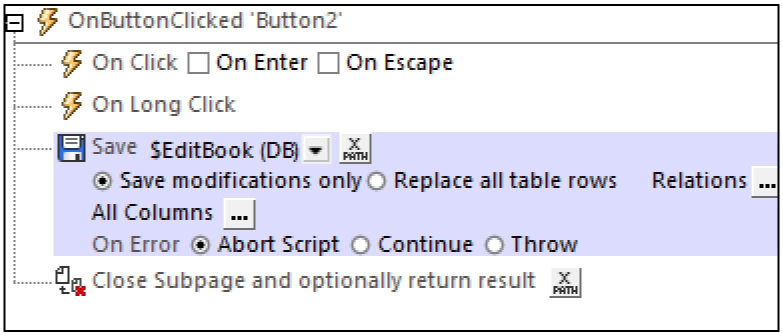
When data is edited on the *Editing Page*, the edited data will be stored in the **`$EditBook`** page source as described in the previous topic, Editing Page: Edit Text and Image Data 141. The user will now want to either save these changes back to the DB or discard them. For these two alternatives, we provide a **Save** button and a **Cancel** button.



| Save | Cancel |
|---|---|
| *Author* | AuthorName ($EditBook) |
| *Website* | Website ($EditBook) |
| *Country* | Country ($EditBook) |
| *Title* | Title ($EditBook) |
| *ISBN* | ISBN ($EditBook) |
| *Publisher* | Publisher ($EditBook) |
| *Print Legth* | NumPages ($EditBook) |
| *Year* | Year ($EditBook) |
| *Genre* | Genre ($EditBook) |
| *Price* | Price ($EditBook) |
| *Book Cover* | Camera · Gallery · Clear |
| Turn Left<br>Turn Right | |

## Save button

The **Save** button uses the Save[803] action (*see screenshot below*) to save the the **`$EditBook`** page source back to its DB source. We save modifications only because we want to update the changed record only. Note that the Save action will automatically update data in both the *Authors* and *Books* tables of our DB. This is because of the hierarchical link between the two tables of the DB[120]. To make sure that changes will be saved to the child *Books* DB table as well, click the **Relations** button and make sure that the option to save modifications has been chosen for the *Books* table.



OnButtonClicked 'Button2'

On Click ☐ On Enter ☐ On Escape

On Long Click

Save $EditBook (DB)

● Save modifications only ○ Replace all table rows Relations ...

All Columns ...

On Error ● Abort Script ○ Continue ○ Throw

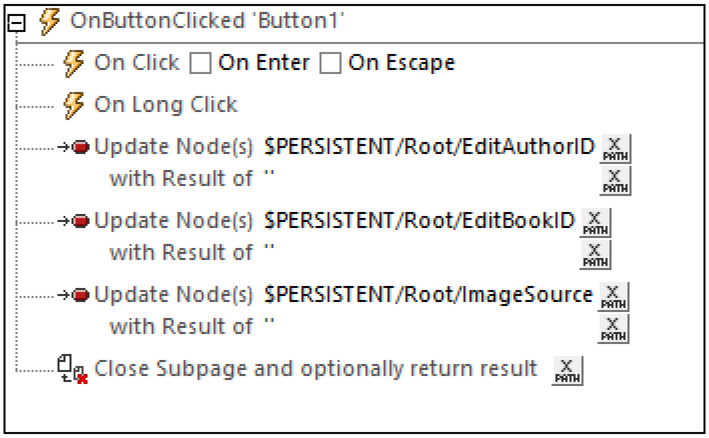
Close Subpage and optionally return result

Since we want the subpage to be closed after the Save[803] action has been executed, we add the Close Subpage[788] action. You might also want to add the clean-up actions described below for the **Cancel** button actions.

### Cancel button

If the changes are to be discarded, all we need to do is to close the *Editing Page* subpage without saving the page source data to the DB. In this way, we return to the main page without having altered the DB. The data on the main page will be that of the unaltered **$BookCatalog** page source.

However, it is best to also clean up any changes that might have been made to the **$PERSISTENT** tree because of user edits. Consequently, we update the relevant **$PERSISTENT** tree odes with the empty string before closing the subpage (*see screenshot below*).



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Update Node(s) $PERSISTENT/Root/EditAuthorID
with Result of ''
Update Node(s) $PERSISTENT/Root/EditBookID
with Result of ''
Update Node(s) $PERSISTENT/Root/ImageSource
with Result of ''
Close Subpage and optionally return result

## 4.4.9 Add New Books

In order to add new books, we can use the *Editing Page*[135] to fill in author and book data. This page uses the **$EditBook** page source, which selects and displays the data of one book: The author ID and book ID of the book to select are obtained from the **$PERSISTENT** tree. So now, if we want to use this subpage as our form for filling data about a new book and display a page with empty entry fields, then we must not select any existing record from the DB and present empty fields for data entry. We can do this by setting the IDs in the **$PERSISTENT** tree to the empty string, respectively, before opening the subpage.

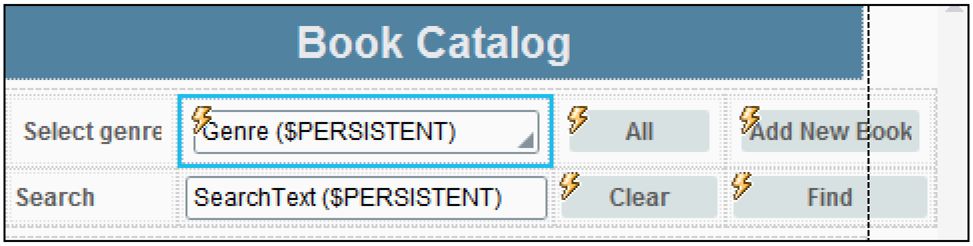
However, we must make sure that we use unique IDs for the primary keys of the new author record and the new book record. We can do this by incrementing each of these values by one over the largest existing ID value in the respective tables.

The text fields and image field of the **$EditBook** page source can be edited in the same way as when modifying the values[141]. When **$EditBook** is saved, the new data will be saved to a new author record and a new book record in the *Authors* and *Books* tables, respectively. Both new records will have not only new unique IDs but also a correct foreign key in the *Books* table—one that links it to the correct author.

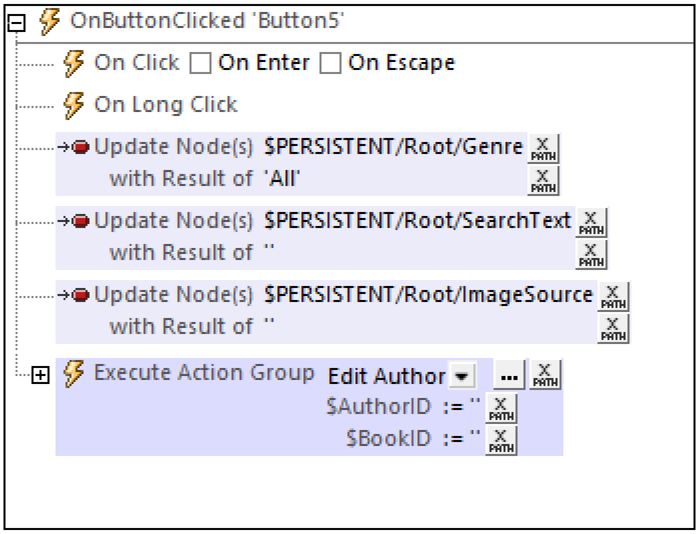
The mechanism described above is implemented through the actions of (i) the **Add New Book** button on the *Main Page*, and (ii) the *OnPageLoad* actions of the *Editing Page*. These two sets of actions are described below.

## Add New Book actions (Main Page)

The actions for going to the *Editing Page* subpage are set on the **Add New Book** button (*see screenshot below*).

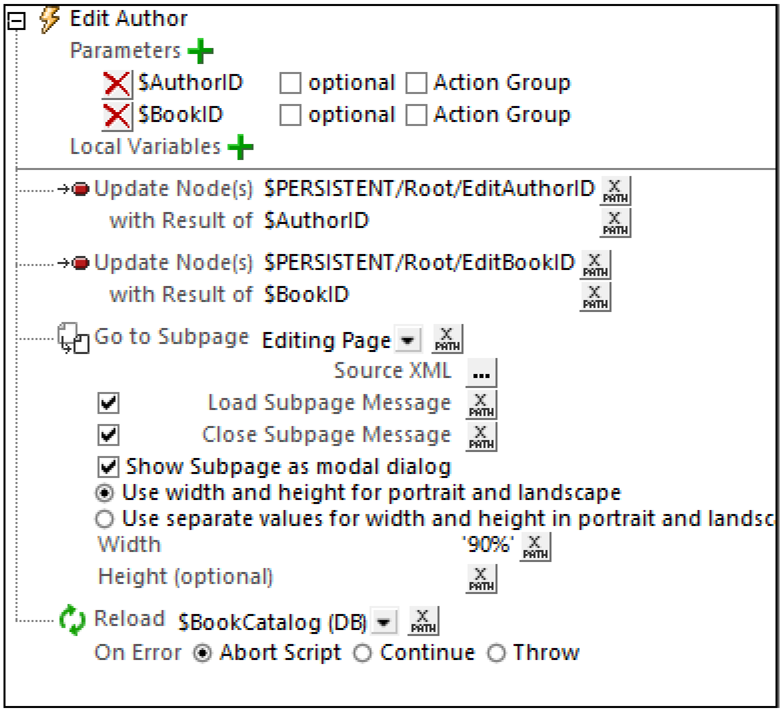


The actions are the same as for the actions for editing a book[133] in that the *Edit Author* Action Group is called. Note, however, that before the call to this Action Group, the data in the **`$PERSISTENT`** tree is reinitialized—with **`Genre`** being set to *All*.



```
OnButtonClicked 'Button5'
  On Click □ On Enter □ On Escape
  On Long Click
  Update Node(s) $PERSISTENT/Root/Genre
    with Result of 'All'
  Update Node(s) $PERSISTENT/Root/SearchText
    with Result of ''
  Update Node(s) $PERSISTENT/Root/ImageSource
    with Result of ''
  Execute Action Group Edit Author
                       $AuthorID := ''
                       $BookID := ''
```

In the call to the *Edit Author* Action Group (*see screenshot above*), the parameters **`$AuthorID`** and **`$BookID`** are set to the empty string. In the *Edit Author* Action Group, the **`EditAuthorID`** and **`EditBookID`** nodes of the **`$PERSISTENT`** tree are updated with these values (*see screenshot below*). This is important because the **`$EditBook`** page source of the *Editing Page* subpage selects the book to display[135] according to the corresponding values in the **`$PERSISTENT`** tree. Since these values are the empty string, there is no record in the DB that will be selected. Consequently, the nodes of the **`$EditBook`** page source would be empty—as will the entry fields on the page.



```
Edit Author
  Parameters
    $AuthorID    □ optional □ Action Group
    $BookID      □ optional □ Action Group
  Local Variables
Update Node(s)  $PERSISTENT/Root/EditAuthorID
  with Result of $AuthorID
Update Node(s)  $PERSISTENT/Root/EditBookID
  with Result of $BookID
Go to Subpage  Editing Page
               Source XML ...
☑     Load Subpage Message
☑     Close Subpage Message
☑ Show Subpage as modal dialog
● Use width and height for portrait and landscape
○ Use separate values for width and height in portrait and landsc
Width                      '90%'
Height (optional)
Reload  $BookCatalog (DB)
On Error ● Abort Script ○ Continue ○ Throw
```

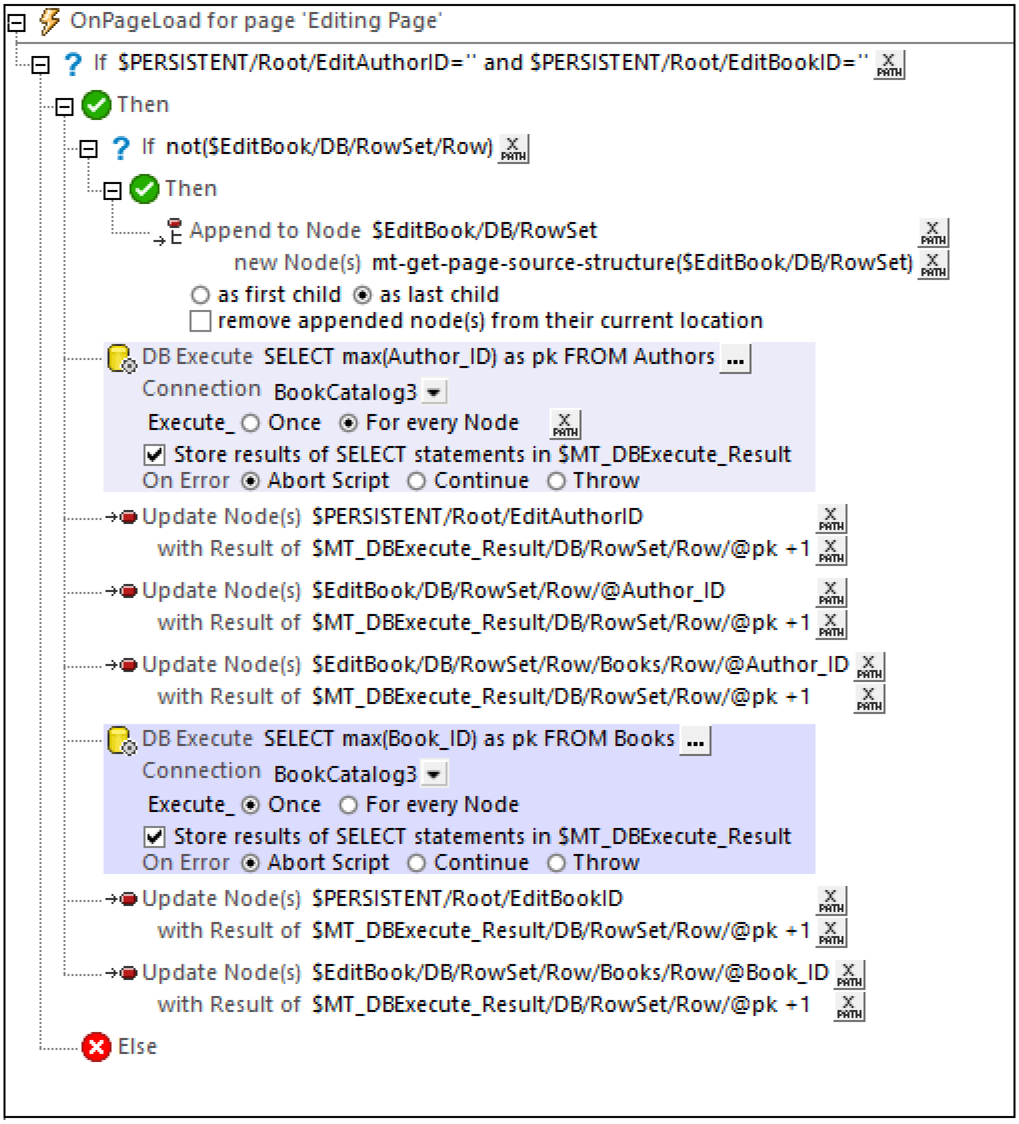
However, there are three DB fields that must not be empty. These are the primary keys of the new *Authors* record and new *Books* record, as well as the foreign key of the *Books* record that relates this *Books* record to the new *Authors* record. The values of these keys are specified as actions of the subpage's OnPageLoad event, as described below

## OnPageLoad actions (Editing Page)

The actions of the OnPageLoad event do the following:

- Ensure that the **`$EditBook`** page source has the expected structure of the **`Authors`** parent, including with the **`Books`** child element. This is done with the **`mt-get-page-source-structure`**[1262] XPath extension function (*see screenshot below*).
- Find out the greatest integer value from among the author IDs. Update three nodes with this value incremented by 1: (i) **`$PERSISTENT/Root/EditAuthorID`**; (ii) the node corresponding to the primary key of the *Authors* table (**`Row/@Author_ID`**); (iii) the node corresponding to the foreign key of the *Books* table (**`Row/Books/Row/@Author_ID`**).
- Find out the greatest integer value from among the book IDs. Update two nodes with this value incremented by 1: (i) **`$PERSISTENT/Root/EditBookID`**; (ii) the node corresponding to the primary key of the *Books* table (**`Row/Books/Row/@Book_ID`**).

**Note:** The **`Author_ID`** and **`Book_ID`** nodes are the primary keys of the *Authors* and *Books* tables, respectively. Each ID must therefore be unique, and this is the reason a new ID is made 1 larger than the greatest of the existing IDs. However, this is not an ideal approach in some situations, and an alternative approach is indicated in the next section below, *Auto-incrementing Primary Keys*.



```
OnPageLoad for page 'Editing Page'
  If $PERSISTENT/Root/EditAuthorID='' and $PERSISTENT/Root/EditBookID=''
    Then
      If not($EditBook/DB/RowSet/Row)
        Then
          Append to Node $EditBook/DB/RowSet
            new Node(s) mt-get-page-source-structure($EditBook/DB/RowSet)
            ○ as first child ● as last child
            □ remove appended node(s) from their current location
      DB Execute SELECT max(Author_ID) as pk FROM Authors ...
        Connection BookCatalog3
        Execute_ ○ Once ● For every Node
        ☑ Store results of SELECT statements in $MT_DBExecute_Result
        On Error ● Abort Script ○ Continue ○ Throw
      Update Node(s) $PERSISTENT/Root/EditAuthorID
        with Result of $MT_DBExecute_Result/DB/RowSet/Row/@pk +1
      Update Node(s) $EditBook/DB/RowSet/Row/@Author_ID
        with Result of $MT_DBExecute_Result/DB/RowSet/Row/@pk +1
      Update Node(s) $EditBook/DB/RowSet/Row/Books/Row/@Author_ID
        with Result of $MT_DBExecute_Result/DB/RowSet/Row/@pk +1
      DB Execute SELECT max(Book_ID) as pk FROM Books ...
        Connection BookCatalog3
        Execute_ ● Once ○ For every Node
        ☑ Store results of SELECT statements in $MT_DBExecute_Result
        On Error ● Abort Script ○ Continue ○ Throw
      Update Node(s) $PERSISTENT/Root/EditBookID
        with Result of $MT_DBExecute_Result/DB/RowSet/Row/@pk +1
      Update Node(s) $EditBook/DB/RowSet/Row/Books/Row/@Book_ID
        with Result of $MT_DBExecute_Result/DB/RowSet/Row/@pk +1
    Else
```

Note the following points:

- The first **IF** expression checks whether the ID nodes in the **$PERSISTENT** tree are both empty. We have set them to be empty when the user chooses to add a new record (*see Add New Book Actions above*).
- The second **IF** expression checks whether any record has been imported from the DB. Since there will be none, the page source structure is appended.
- The first of the two DB Execute 861 actions (*highlighted blue*) queries the DB for the greatest integer value from among the author IDs and saves the value as an attribute named **@pk** in the **$MT_DBExecute_Result** 1304 dynamic variable. (This variable is an XML tree; it can be serialized with the XPath function **serialize** if you want to see its structure)
- The first set of Update Node 886 actions is used to update the author ID nodes in the **$PERSISTENT** tree and the **$EditBook** page source.

- The second of the two DB Execute[861] actions is similar to the first. It queries the DB for the greatest integer value from among the book IDs and saves the value as an attribute named **`@pk`** in the **`$MT_DBExecute_Result`**[1304] dynamic variable.
- The second set of Update Node[886] actions is used to update the book ID nodes in the **`$PERSISTENT`** tree and the **`$EditBook`** page source.

The record now has its important ID fields filled with the relevant unique IDs, and it is now displayed for data entry in the *Editing Page* subpage. The data that is entered is saved in the **`$EditBook`** page source. On saving the page source, the new data is added as an *Authors* record and a *Books* record to the DB, related to each other by the *Books* record's foreign key. The saving and canceling actions[145] are the same as those used when editing a record.

### Auto-incrementing Primary Keys

In our example above, we have calculated the value of a new record's primary key in the following way: We find the greatest integer among all the IDs of the respective table, add 1 to this value, and assign the resulting value as the ID of the new record.

However, this approach would not be ideal if, while one user is adding a new record and has not yet saved it, a second user starts to add a new record. In this case, both new records would have the same ID, the uniqueness criterion would not be fulfilled, and, consequently, one of the records would be rejected by the DB.

An alternative approach would be to define the primary key fields of the DB to auto-increment when a new record is added to the DB. There would then be no need to enter a value for the primary key field, let alone a unique value. This is because a unique value will be entered automatically by the DB when the record is entered.

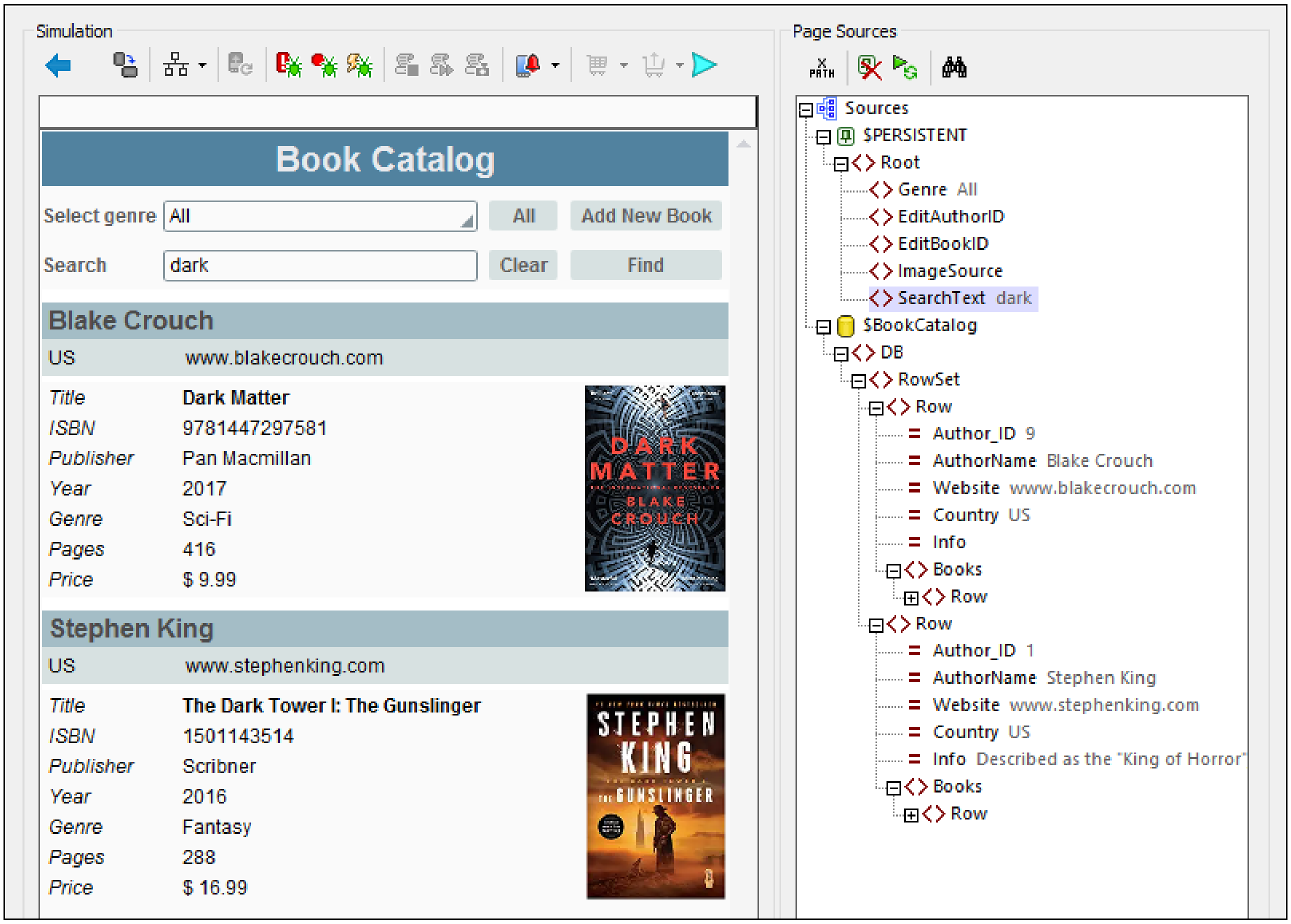
You can set a DB field to auto-increment in one of the following ways:

- In the DB, define the field to be auto-incrementing. When a new record is saved from the solution to the DB, the primary key field will be automatically incremented with a unique value.
- In the Page Sources Pane[270] of your design, right-click the page source field you want to auto-increment and select **DB Field | Is Auto Increment**. When the record is saved to the DB, the field will be auto-incremented.

## 4.4.10 Search the DB

The goal of the search is to filter the display of records so that it displays only books where the search string occurs in any of a book's or an author's significant text fields. If the search string occurs in a field of an *Authors* record—as opposed to a field in a *Books* record—then all the books of that author should be displayed.

The screenshot below shows the results of a search for the string "dark". The search finds two books. In both cases the search string occurs in the book's *Title* field.



Book Catalog

Select genre: All

Search: dark

**Blake Crouch**

US www.blakecrouch.com

| | |
|---|---|
| *Title* | **Dark Matter** |
| *ISBN* | 9781447297581 |
| *Publisher* | Pan Macmillan |
| *Year* | 2017 |
| *Genre* | Sci-Fi |
| *Pages* | 416 |
| *Price* | $ 9.99 |

**Stephen King**

US www.stephenking.com

| | |
|---|---|
| *Title* | **The Dark Tower I: The Gunslinger** |
| *ISBN* | 1501143514 |
| *Publisher* | Scribner |
| *Year* | 2016 |
| *Genre* | Fantasy |
| *Pages* | 288 |
| *Price* | $ 16.99 |

```
Sources
  $PERSISTENT
    Root
      Genre  All
      EditAuthorID
      EditBookID
      ImageSource
      SearchText  dark
  $BookCatalog
    DB
      RowSet
        Row
          = Author_ID  9
          = AuthorName  Blake Crouch
          = Website  www.blakecrouch.com
          = Country  US
          = Info
          Books
            Row
        Row
          = Author_ID  1
          = AuthorName  Stephen King
          = Website  www.stephenking.com
          = Country  US
          = Info  Described as the "King of Horror"
          Books
            Row
```

## Local search versus DB search

The search can be carried out locally, on the mobile device, in the data that has been downloaded from the DB, or it can be carried out in the DB on the server. Each mechanism has relative advantages over the other, as explained below.

- If the search is carried out locally, all DB records must be loaded into memory and searched. This mechanism is faster than if the search is carried out in the DB, but it consumes more memory since all the records must be held in memory, especially if the amount of record data is large.
- If the search is carried out by sending an SQL request to the DB, the time taken for transactions with the DB will be more. However, since only, the searched records are returned, memory consumption for data storage will be lower.

You should weigh up the relative merits when deciding which mechanism is better suited for a particular case.

## Mechanism

It is possible to design different approaches to the search. The mechanism we have used combines an SQL DB search with a local MobileTogether-specific approach, and it is outlined broadly below.

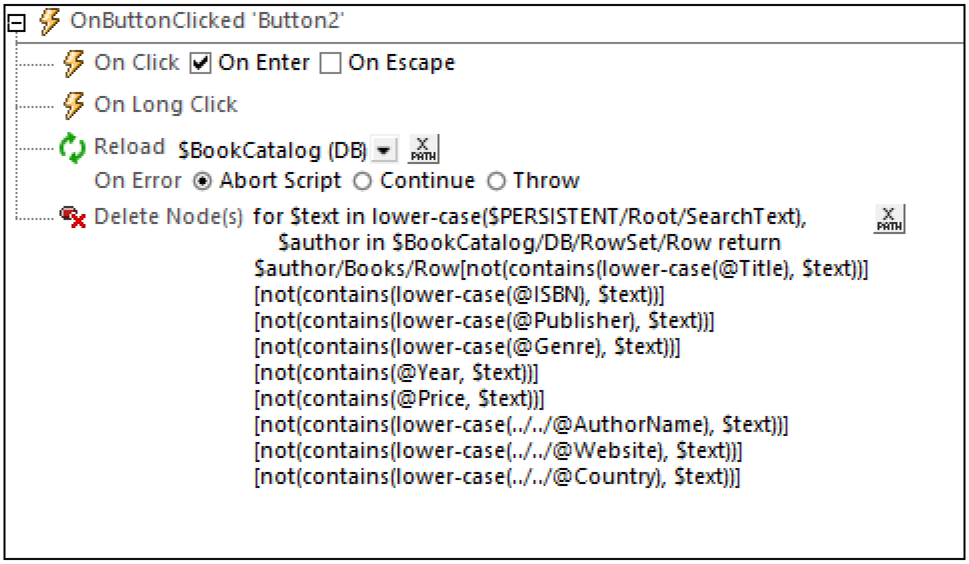
1. The search string is entered in an Edit Field[495].
2. Since the Edit Field[495] is associated with the **`SearchText`** node of the **`$PERSISTENT`** page source, the search text will automatically be passed to this node as the search text is being entered in the Edit Field[495] (*see highlighted node in screenshot above*)
3. On clicking the **Find** button, two actions are executed in sequence: (i) the **`$BookCatalog`** page source is reloaded[801] with only those *Authors* records that contain the search string in the data of either the author or any book child (*see Point 4 below*); (ii) a Delete Node(s)[879] action removes those books (of the selected authors) that do not contain the search string in any of its fields, leaving only the books of that author that contain the search string (*see Point 5 below*).
4. The reload of the page source[801] triggers a search in the DB for the search string via an SQL statement. As a result, the **`$BookCatalog`** page source will be reloaded with only the relevant authors (where the search string occurs either in the author data or in the data of at least one of the author's books). See the section *SQL Search of the DB* below for details.
5. Since the SQL search of the DB returns authors with all their books, the books where the search string does not occur are deleted. See the section *Removing Non-matched Books* below for details.

## The search string

An Edit Field[495] has been created in which the user can enter the search string. The Edit Field[495].is associated with the **`$PERSISTENT/Root/SearchText`** node (by dragging-and-dropping the node onto the Edit Field[495]). As a result, the search text will be stored in this node, from where it can be accessed subsequently for actions.

## The Find button

The search is started when the **Find** button is clicked. The actions of the button's OnClicked event (*screenshot below*) are (i) a Reload[801] of the **`$BookCatalog`** page source and (ii) a Delete Node(s)[879] action. Both actions are described in detail below. For the moment, it is important to know that the Reload[801] action for the **`$BookCatalog`** page source starts the search in the DB. See the next section, *SQL Search of the DB*. The Delete Node(s)[879] action is described in the section *Removing Non-matched Books* below



OnButtonClicked 'Button2'

On Click ☑ On Enter □ On Escape

On Long Click

Reload $BookCatalog (DB)

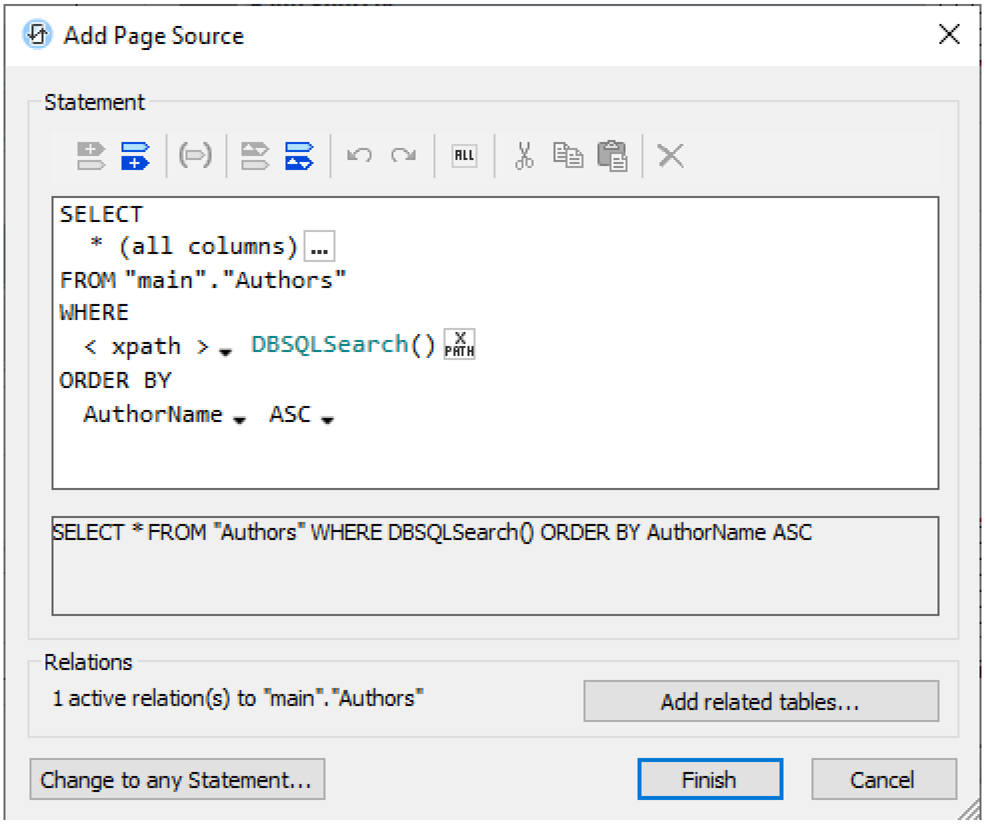
On Error ● Abort Script ○ Continue ○ Throw

Delete Node(s)
```
for $text in lower-case($PERSISTENT/Root/SearchText),
    $author in $BookCatalog/DB/RowSet/Row return
$author/Books/Row[not(contains(lower-case(@Title), $text))]
[not(contains(lower-case(@ISBN), $text))]
[not(contains(lower-case(@Publisher), $text))]
[not(contains(lower-case(@Genre), $text))]
[not(contains(@Year, $text))]
[not(contains(@Price, $text))]
[not(contains(lower-case(../../@AuthorName), $text))]
[not(contains(lower-case(../../@Website), $text))]
[not(contains(lower-case(../../@Country), $text))]
```

## SQL search of the DB

When the Reload[801] action for **`$BookCatalog`** is triggered, an SQL statement—in the data-selection definition of the DB page source—selects every row of the *Authors* table that has any field containing the search text (either in the *Authors* record itself or in any of the record's related *Books* records). This is achieved through the following steps.

*$BookCatalog page source configuration*
The **`$BookCatalog`** page source was originally configured to select all the records of the Authors table[126], with each *Authors* record containing related *Books* records as children. Now we add a **`WHERE`** clause to the SQL statement in order to filter the *Authors* records to select only those records where the search term occurs. Since the **`WHERE`** clause needs to be constructed with a complex XPath expression, we will use a function to implement the XPath expression. The function is called **`DBSQLSearch()`**, as shown in the screenshot below. In the screenshot, you can read the SQL statement in the pane near the bottom of the window. In this SQL statement, the **`WHERE`** clause will be the return value of the **`DBSQLSearch()`** function.


Add Page Source
Statement

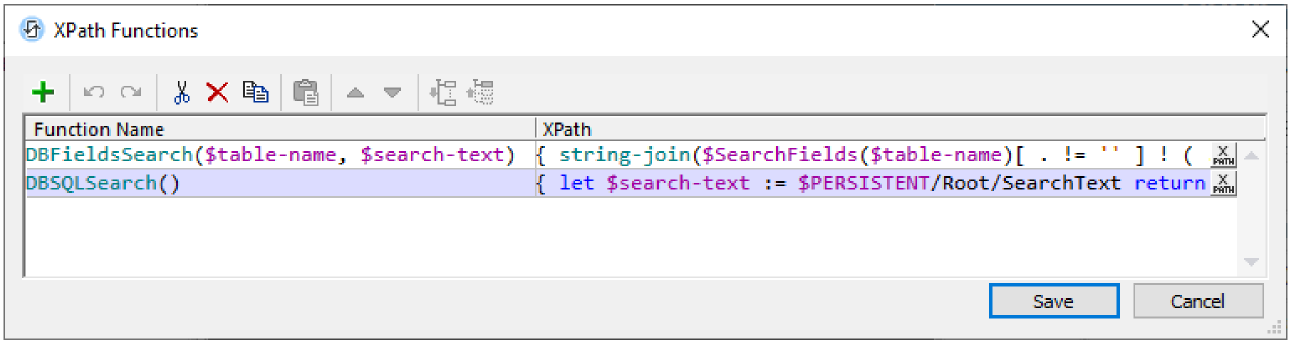
```
SELECT
  * (all columns) ...
FROM "main"."Authors"
WHERE
  < xpath > DBSQLSearch()
ORDER BY
  AuthorName  ASC
```

```
SELECT * FROM "Authors" WHERE DBSQLSearch() ORDER BY AuthorName ASC
```

Relations
1 active relation(s) to "main"."Authors"
Add related tables...
Change to any Statement...
Finish
Cancel

**Note:** To edit or view the configuration of the **`$BookCatalog`** page source, click the *Configuration* icon of **`$BookCatalog`** in the Page Sources Pane[270].

*Functions for the search*
For the search, we create two functions: **`DBSQLSearch()`** and **`DBFieldsSearch()`**. The XPath Functions dialog to create these functions is accessed via the menu command **Project | XPath/XQuery Functions**[1591] and it is shown in the screenshot below.



| Function Name | XPath |
|---|---|
| DBFieldsSearch($table-name, $search-text) | { string-join($SearchFields($table-name)[ . != '' ] ! ( |
| DBSQLSearch() | { let $search-text := $PERSISTENT/Root/SearchText return |

⊟ **DBSQLSearch() function**

```
declare function DBSQLSearch()
{
   let $search-text := $PERSISTENT/Root/SearchText return
   if ($search-text != '' ) then
   DBFieldsSearch( 'Authors', $search-text ) ||
   ' OR Author_ID IN (SELECT AuthorID FROM Books WHERE ' ||
            DBFieldsSearch( 'Books', $search-text ) || ')'
   else
      ''
}
```

⊟ **DBFieldsSearch() function**

```
declare function DBFieldsSearch($table-name, $search-text)
{
   string-join($SearchFields($table-name) ! ( . || ' LIKE ''%' || $search-text ||
  '%''' ), ' OR ' )
}
```

⊟ **$SearchFields variable**

```
map {
   'Authors'    : ('AuthorName', 'Website', 'Country'),
   'Books'      : ('Title', 'ISBN', 'Publisher', 'Year', 'Genre', 'Price')
}
```

The **DBSQLSearch()** function does the following:

- If the search text is not the empty string, then a **SELECT** statement is built using the **DBFieldsSearch()** function twice: the first time to search the fields of the *Authors* table, the second time to search the fields of the *Books* table.
- The **DBFieldsSearch()** function builds the **WHERE** clause to search each table (*Authors* and *Books*) by referencing the **$SearchFields** map of each DB table's columns. The **$SearchFields** map is stored as a user-defined variable that is accessed via the menu command **Project | Global Variables**1590.

- Else, as defined in the **`DBSQLSearch()`** function, if the search text is the empty string, then the **`WHERE`** clause of the SQL statement for **`$BookCatalog`** will be the empty string (*see the DBSQLSearch() function above*). Effectively, in this case, the SQL statement will have no **`WHERE`** clause, and all the *Authors* records will be returned.

The structure of the **`SELECT`** statement created when the search string is not empty is as follows:

```
SELECT <AuthorsFields-1 to AuthorsFields-Z> FROM "Authors"
WHERE AF1 LIKE '%SearchText%' OR AF2 LIKE '%SearchText%' ...  OR AFZ LIKE '%SearchText
%'
OR Author_ID IN (SELECT AuthorID FROM Books
WHERE BF1 LIKE '%SearchText%' OR BF2 LIKE '%SearchText%' ...  OR BFZ LIKE '%SearchText
%')
```

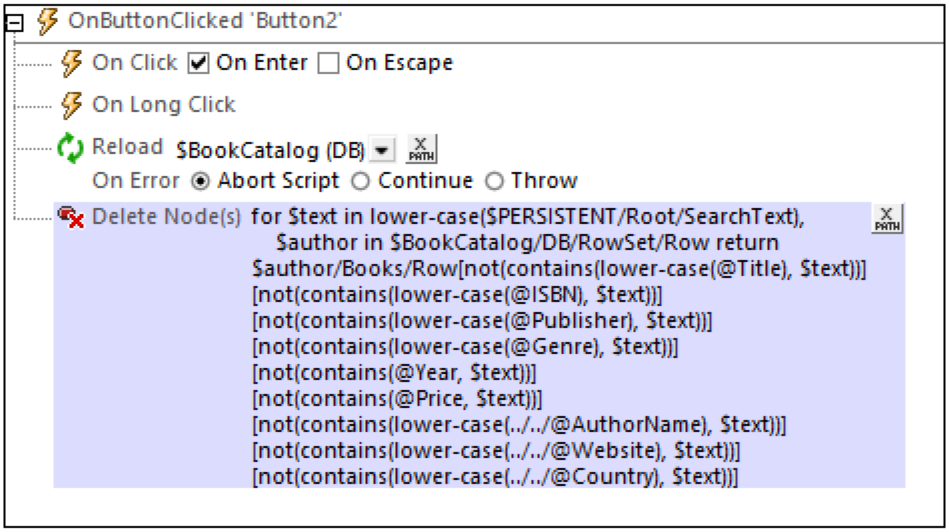
This expression will select *Authors* records that have either (i) any *Authors* filed that matches the search text or (ii) any related *Books* record that has a field that matches the search text. Note that if the search in the related *Books* records returns a match, then it is the parent *Authors* record that is selected.

It is important to note that, since, eventually, it is *Authors* records that are selected, these *Authors* records will be selected with all their related *Books* records—even if one or more of these *Books* records do not contain the search text. How to display only those books that contain the search text is described in the next section, *Removing Non-matched Books*.

## Removing non-matched books

Since selected *Authors* records are returned with **all** their related (child) *Books* records (*see discussion above*), the following possibilities arise:

- The search text is found in the *Authors* table. In this case, we can show the author's details together with all the related *Books* records of that author.
- The search text is found in one of the related *Books* tables of an *Authors* record. In this case, we should show the author's details and details of the matched book/s only. One way to do this would be to remove the non-matched books from the **`$BookCatalog`** page source. In our example, we have done this by adding a Delete Node(s) 879 action to the **Find** button after the page source has been reloaded (*see screenshot below*).



⊟ OnButtonClicked 'Button2'
- On Click ☑ On Enter □ On Escape
- On Long Click
- Reload $BookCatalog (DB)
  On Error ● Abort Script ○ Continue ○ Throw
- Delete Node(s) for $text in lower-case($PERSISTENT/Root/SearchText),
  $author in $BookCatalog/DB/RowSet/Row return
  $author/Books/Row[not(contains(lower-case(@Title), $text))]
  [not(contains(lower-case(@ISBN), $text))]
  [not(contains(lower-case(@Publisher), $text))]
  [not(contains(lower-case(@Genre), $text))]
  [not(contains(@Year, $text))]
  [not(contains(@Price, $text))]
  [not(contains(lower-case(../../@AuthorName), $text))]
  [not(contains(lower-case(../../@Website), $text))]
  [not(contains(lower-case(../../@Country), $text))]

⊟ XPath expression of the Delete Nodes action

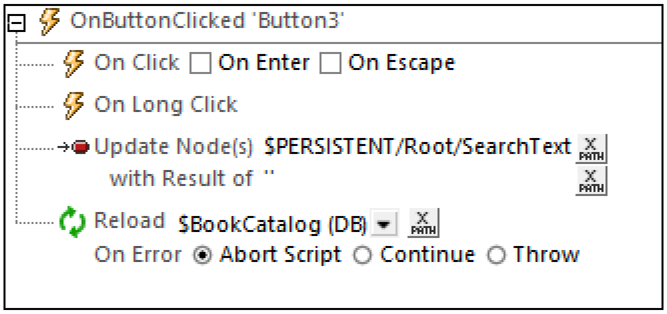
```
for $text in lower-case($PERSISTENT/Root/SearchText),
    $author in $BookCatalog/DB/RowSet/Row
return $author/Books/Row
[not(contains(lower-case(@Title), $text))]
[not(contains(lower-case(@ISBN), $text))]
[not(contains(lower-case(@Publisher), $text))]
[not(contains(lower-case(@Genre), $text))]
[not(contains(@Year, $text))]
[not(contains(@Price, $text))]
[not(contains(lower-case(../../@AuthorName), $text))]
[not(contains(lower-case(../../@Website), $text))]
[not(contains(lower-case(../../@Country), $text))]
```

The XPath expression of the Delete Node(s) 879 action works as follows:

- The **$text** variable contains a lower-cased variant of the search string. This will enable the search to be case-insensitive.
- The nodes that are returned for deletion will be selected from the descendant *Books* records of the current author.
- The books that are selected for deletion must not contain the search string in any of the significant *Books* fields. This is specified by building a sequence of predicate filters; each predicate is inside square brackets. All of the predicates must evaluate to **true** for the book to qualify for deletion. If one predicate evaluates to **false** (which would happen if the search string exists in the field being checked in that predicate), then the current *Books* record is not selected for deletion, and the next book is checked.
- Note that the predicate filters not only check the fields of the *Books* records but also the fields of the parent *Authors* record (see the last three predicates).

## Clear the search string

After a search has been carried out, the **`$BookCatalog`** page source will contain only the *Authors* and *Books* records that were returned by the search mechanism described above. The **Clear** button (*see first screenshot of this topic*) clears the search string and reloads the **`$BookCatalog`** page source so that it contains all the *Authors* records. The actions of the **Clear** button are shown in the screenshot below.



```
⊟ OnButtonClicked 'Button3'
    On Click □ On Enter □ On Escape
    On Long Click
    Update Node(s) $PERSISTENT/Root/SearchText
      with Result of ''
    Reload $BookCatalog (DB)
    On Error ● Abort Script ○ Continue ○ Throw
```

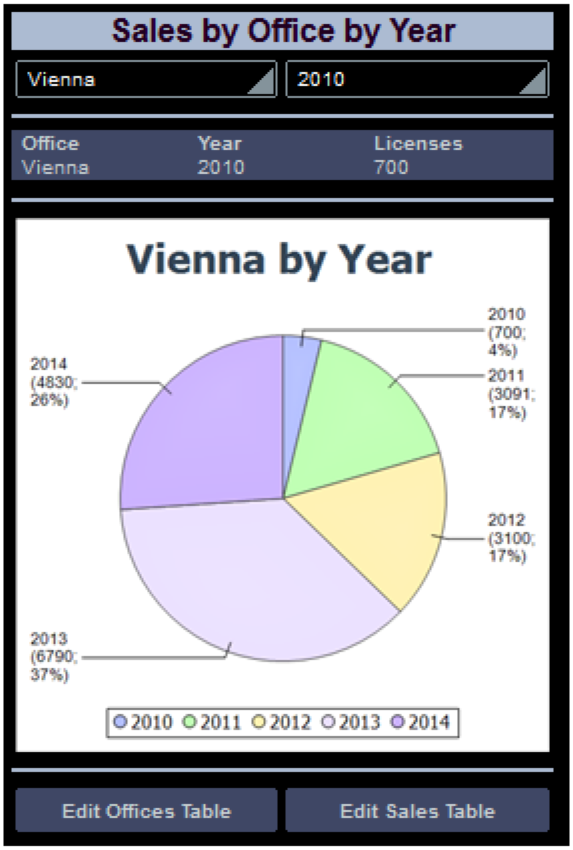
The Update Node(s)[886] action changes the **`$PERSISTENT/Root/SearchText`** node to contain the empty string. Since this node is associated with the Edit Field[495] in which the search text is entered, the empty string value is displayed in the Edit Field[495], effectively clearing the Edit Field[495] (*see the section "The Search String" above*). Reloading the **`$BookCatalog`** page source with the **`$PERSISTENT/Root/SearchText`** nodeset to the empty string loads all *Authors* records into the page source (see the description of the **`DBSQLSearch()`** function above).

## 4.5 Database-And-Charts

This Database-And-Charts tutorial (`DBAndCharts.mtd`) shows you how to work with databases (DBs) and charts. It explains how to:

- Set up DB tables as page sources so that DB table data can be displayed and edited
- How to display DB data based on end user choices
- How to create charts that are based on the DB data

The screenshot below shows the first page of the `DBAndCharts.mtd` solution. The end user can select the office and year for which sales are to be displayed. These selections are made in combo boxes at the top of the design. The total sales for that year are then displayed in the *Licenses* column of the results table below the combo boxes. Whenever a new office or year is selected in either of the combo box, the results table is automatically updated to reflect the new selection. Additionally, a pie chart showing the sales, by year, for the selected office is displayed. Each slice represents a year, with its sales volume and the percentage of all sales till now that year represents. Whenever a new office is selected, a pie chart showing the statistics of that particular office is displayed.



Sales by Office by Year

Vienna 2010

| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

Below the chart are two buttons that each take you to a page in which the relevant DB table data can be edited.

## The tutorial files

The files for this tutorial are located in your (72 My) Documents 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\DBAndCharts`**.

- The design file is `DBAndCharts.mtd`. Open it and read the descriptions in the tutorial to see how the design was built and how it works.
- The MS Access database, `OfficeSales_DB.mdb`, contains the tables used as the page sources of the design.

## Video demo of how to create charts

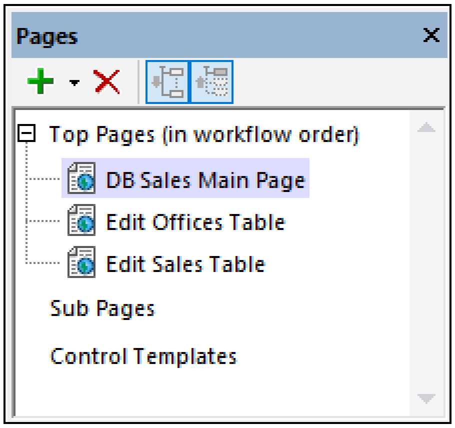
The Altova website provides a video demo that shows how to create charts in your solution.

# 4.5.1 The Project Structure

Open `DBAndCharts.mtd` 159 and validate it (**Project | Validate**) to check that the file correctly connects to the Access DB, `OfficeSales_DB.mdb`. If there is a connection error, make sure you correct this before proceeding. (See the section, Page Sources of the Main Page 163, for more information.)

As shown in the Pages Pane 257 (*screenshot below*), the project consists of three top pages:

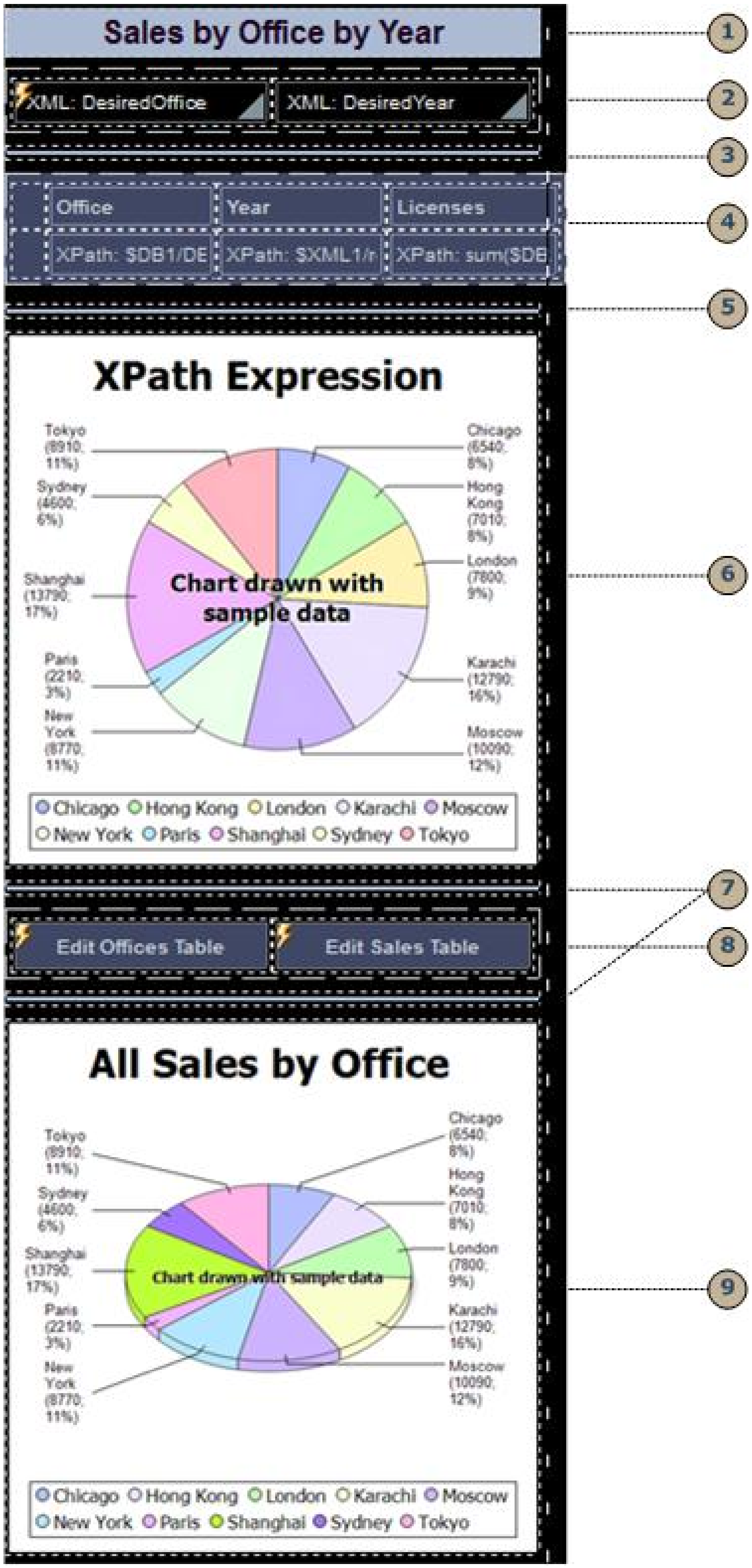
- *DB Sales Main Page:* This is the start page. It displays the DB data and has two buttons that go, respectively, to the other two top pages.
- *Edit Offices Table:* Is arrived at via a button click from the main page and enables editing of the DB's Offices table.
- *Edit Sales Table:* Is arrived at via a button click from the main page and enables editing of the DB's Sales table.



When the solution runs, note that it is the first top page in the list above, *DB Sales Main Page*, that is loaded in the client app.

### 4.5.2 The Main Page

The design of the *DB Sales Main Page* is show below. Its components are numbered in the callouts and are described below.


Sales by Office by Year
XML: DesiredOffice
XML: DesiredYear
Office
Year
Licenses
XPath: $DB1/DE
XPath: $XML1/r
XPath: sum($DB
XPath Expression
Tokyo (8910; 11%)
Chicago (6540; 8%)
Sydney (4600; 6%)
Hong Kong (7010; 8%)
London (7800; 9%)
Shanghai (13790; 17%)
Chart drawn with sample data
Paris (2210; 3%)
Karachi (12790; 16%)
New York (8770; 11%)
Moscow (10090; 12%)
Chicago Hong Kong London Karachi Moscow
New York Paris Shanghai Sydney Tokyo
Edit Offices Table
Edit Sales Table
All Sales by Office
Tokyo (8910; 11%)
Chicago (6540; 8%)
Sydney (4600; 6%)
Hong Kong (7010; 8%)
Shanghai (13790; 17%)
London (7800; 9%)
Chart drawn with sample data
Paris (2210; 3%)
Karachi (12790; 16%)
New York (8770; 11%)
Moscow (10090; 12%)
Chicago Hong Kong London Karachi Moscow
New York Paris Shanghai Sydney Tokyo
1
2
3
4
5
6
7
8
9

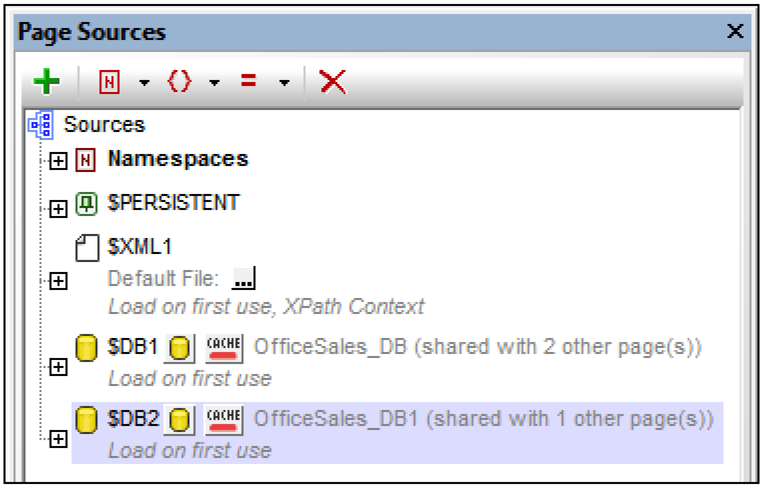
All the components are controls that have been dragged from the Controls Pane[266] and dropped into the design. Each has then been assigned properties in the Styles & Properties Pane[274]. For controls that need to be associated with data from the page sources, the appropriate source node has been assigned by dragging the page source node onto the control. The combo boxes and buttons additionally have actions associated with their events. Actions are assigned in the Actions dialog for the control, which is accessed by right-clicking the control and selecting the **Control Actions for...** command.

| | |
|---|---|
| 1 | A label control to display the title of the page; style properties applied |
| 2 | Combo boxes for end user selection of `Office` and `Year`. *See detailed description*[167] |
| 3 5 7 | A horizontal line control as layout component; style properties applied |
| 4 | Table control with cells that contain DB data. *See detailed description*[169] |
| 6 9 | Chart controls that display DB data in the form of charts. *See detailed description*[170] |
| 8 | Button controls with `OnButtonClicked` actions that go to *Edit Offices*[175] and *Edit Sales*[181] pages |

The *DB Sales Main Page* has an action defined for its `OnPageLoad` event (**Page | Page Actions**) that updates a page source node. This action is explained in the next section, Page Sources of the Main Page[163].

## 4.5.3 Page Sources of the Main Page

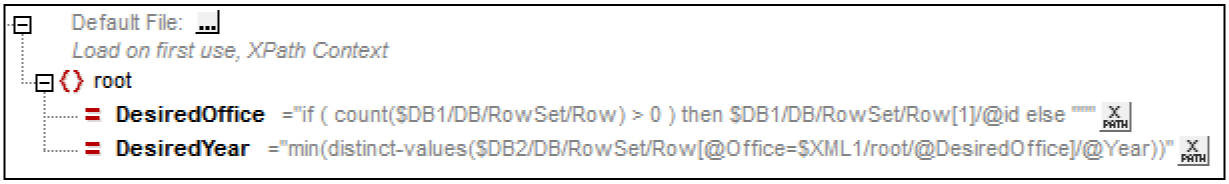
The main page has three page sources: `$XML1`, `$DB1`, and `$DB2`. These are displayed and managed in the Page Sources Pane[270] (*see screenshot below*).


Page Sources
Sources
Namespaces
$PERSISTENT
$XML1
Default File:
Load on first use, XPath Context
$DB1
OfficeSales_DB (shared with 2 other page(s))
Load on first use
$DB2
OfficeSales_DB1 (shared with 1 other page(s))
Load on first use

The XPath context node of the page is the root node **`$XML1`**. This means that all XPath expressions on this page have `$XML1` as their context node. To locate a node in any of the other trees (`$DB1` and `$DB2`, which are the root nodes of these trees) start the XPath locator path with the respective root node.

## The first page source: $XML1

This page source was created as an editable empty XML. The root node **`$XML1`** contains a root element (**`root`**), which has two attributes (**`DesiredOffice`** and **`DesiredYear`**). The root node, `$XML1`, was set (via its context menu) as the XPath context node for the page two. No default file is set, so no data is imported into the tree.

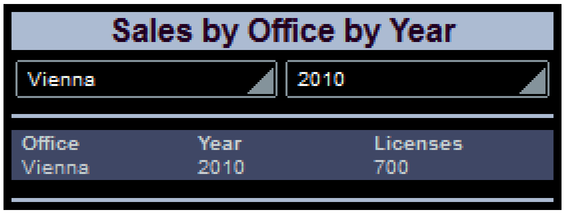


```
Default File: ...
Load on first use, XPath Context
root
    DesiredOffice ="if ( count($DB1/DB/RowSet/Row) > 0 ) then $DB1/DB/RowSet/Row[1]/@id else """
    DesiredYear ="min(distinct-values($DB2/DB/RowSet/Row[@Office=$XML1/root/@DesiredOffice]/@Year))"
```

This page source (`$XML1`) has been created to hold the end user's combo box selections[167]:

- The `DesiredOffice` attribute holds the end user's `Office` selection
- The `DesiredYear` attribute holds the end user's `Year` selection

In order to hold the data selected in the combo boxes, the two attribute nodes are associated with the combo boxes as page source links. Each of the two page source links is made by dragging the attribute node onto the respective comb box (*see simulator screenshot below*).



Sales by Office by Year

Vienna | 2010

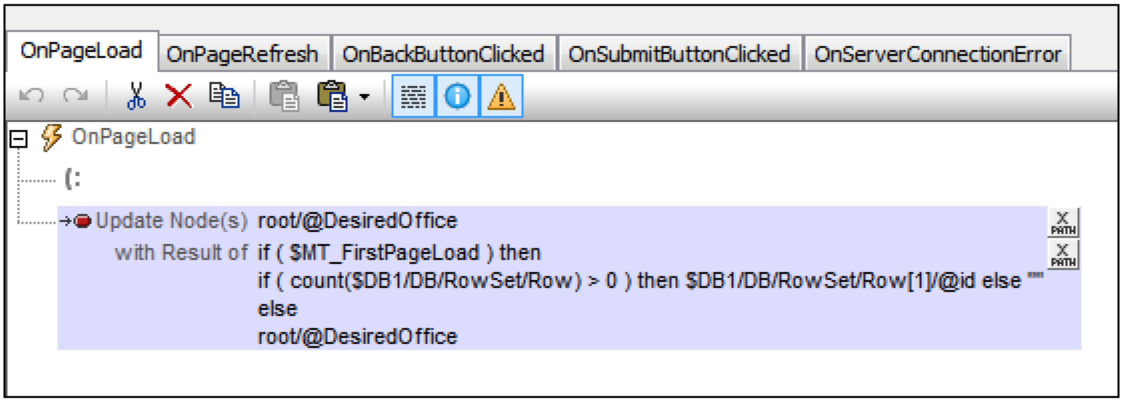
| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

Each of the nodes has been given an initial value when the page loads (via the context menu command **Ensure Exists on Load (XPath Value)**). This is because the value of the node appears in the associated combo box, and we want the combo box to have an initial selection (*see simulator screenshot above*). The XPath expressions that provide the initial values are:

- For `@DesiredOffice`: `if (count($DB1/DB/RowSet/Row) > 0) then $DB1/DB/RowSet/Row[1]/@id else ""`
  *If there is one or more records in `$DB1`, sets the `@id` value of the first record as the value of `@DesiredOffice`. If there is no record, sets the empty string as the value of `@DesiredOffice`.*

- For `@DesiredYear`: `min(distinct-values($DB2/DB/RowSet/Row[@Office=$XML1/root/@DesiredOffice]/@Year))`
  *In `$DB2`, selects all the records of the office selected in `@DesiredOffice`, collects the unique years from these records, and then selects the year with the minimum numerical value.*

Additionally, we have specified that the **`@DesiredOffice`** node is correctly filled whenever the main page loads. This is done with an *Update Node* action on the **`OnPageLoad`** event of the main page (**Page | Page Actions**).

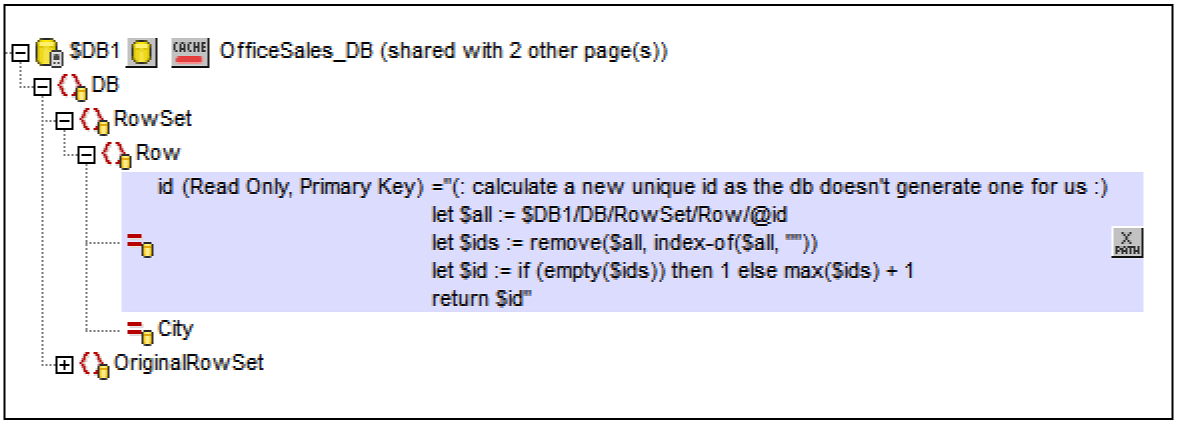


```
OnPageLoad
  (:
  Update Node(s) root/@DesiredOffice
  with Result of if ( $MT_FirstPageLoad ) then
                 if ( count($DB1/DB/RowSet/Row) > 0 ) then $DB1/DB/RowSet/Row[1]/@id else ""
                 else
                 root/@DesiredOffice
```

The action updates the **`@DesiredOffice`** node. If this is the first loading of the page, then the ID of the first office is passed as the content of **`@DesiredOffice`**. Otherwise the value is what is already present in **`@DesiredOffice`**. The result of this is that during an execution, the value in **`@DesiredOffice`** is not changed, but the value is initialized whenever the page is loaded for the first time.

## The second page source: $DB1

The second page source (**`$DB1`**) is the `Offices` table in the MS Access database, `OfficeSales_DB.mdb` 159. The data for this page source comes from the DB's `Office` table.



```
$DB1  OfficeSales_DB (shared with 2 other page(s))
  DB
    RowSet
      Row
        id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                                      let $all := $DB1/DB/RowSet/Row/@id
                                      let $ids := remove($all, index-of($all, ""))
                                      let $id := if (empty($ids)) then 1 else max($ids) + 1
                                      return $id"
        City
    OriginalRowSet
```

The Offices table has two columns (**`id`** and **`City`**), which are represented in the data tree as attributes of the `Row` element, which itself corresponds to a row in the DB table. Since the `id` column is the primary key and values in it cannot be changed, we cannot edit this column. However, we need to create `id` values for new rows. We automate this by writing an XQuery expression to generate the `id` value for every new row that is created. The XQuery expression is inserted by using the context menu command, **Ensure Exists on Load (XPath Value)**:

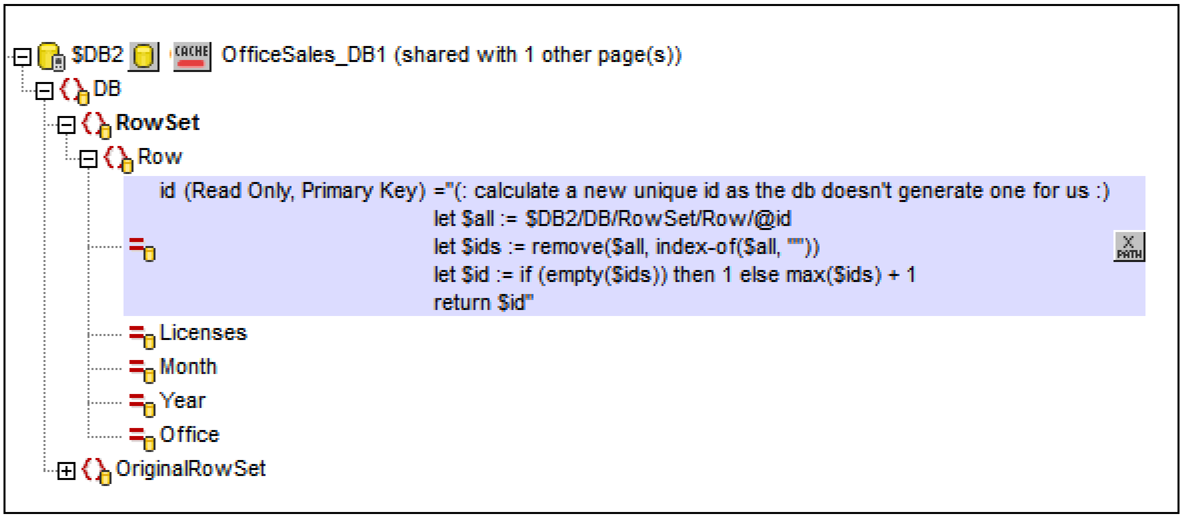
```
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```

Note that the **id** value is the unique ID number of the office, whereas the **City** value is the name of the city in which the office is. This is important because while it is the **id** that is used to uniquely identify an office (via the `$XML1/root/@DesiredOffice` node), it is the name of the city that we use to identify an office to the end user.

An **OriginalRowSet** node must be created (via the context menu) if any node in the page source is to be edited. This is required so that `OriginalRowSet` holds the original data while **RowSet** holds the current (edited) data. The two sets of data (original and edited) are required so that MobileTogether Designer can tell the difference between what is new, updated, and deleted, and can make the necessary changes at the right time. It is also required so that it can create new primary keys with the XQuery **let** statement. When the database is updated, the updated data becomes the new original data and is entered in the `OriginalRowSet` node.

## The third page source: $DB2

The third page source (**$DB2**) is the Sales table in the MS Access database, `OfficeSales_DB1.mdb` 159. The data for this data tree comes from the DB's Sales table.



$DB2 OfficeSales_DB1 (shared with 1 other page(s))
DB
RowSet
Row
id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)

```
let $all := $DB2/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id"
```

Licenses
Month
Year
Office
OriginalRowSet

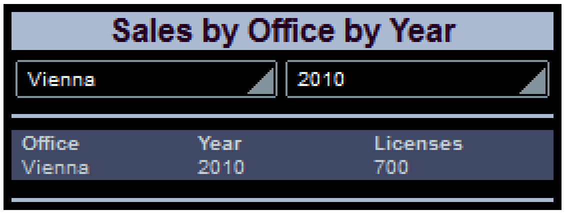
Each row in the Sales table has five columns (**id** , **Licenses**, **Month**, **Year**, and **Office**). The DB table row corresponds to the **Row** element in the page source tree. The table's columns correspond to the attributes of the **Row** element. The `id` attribute has an XQuery expression to generate the `id` value for every new row that is created. The XQuery expression is inserted by using the context menu command, **Ensure Exists before Page Load (XPath Value)**:

```
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```

An **OriginalRowSet** node must be created (via the context menu) if any node in the page source is to be edited. This is required so that `OriginalRowSet` holds the original data while **RowSet** holds the current (edited) data.

## 4.5.4 The Combo Boxes

The two combo boxes at the top of the page are used to accept end user selections and to display data based on these selections. The screenshot below shows the combo boxes when the solution is run; the tabular report below the combo boxes is based on the combo box selections.



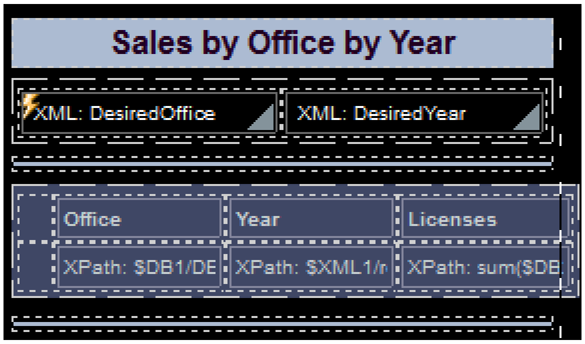
Sales by Office by Year

Vienna

2010

| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

The next screenshot shows the combo boxes in the design. The combo boxes have been placed in separate cells of a table for layout purposes.



Sales by Office by Year

XML: DesiredOffice

XML: DesiredYear

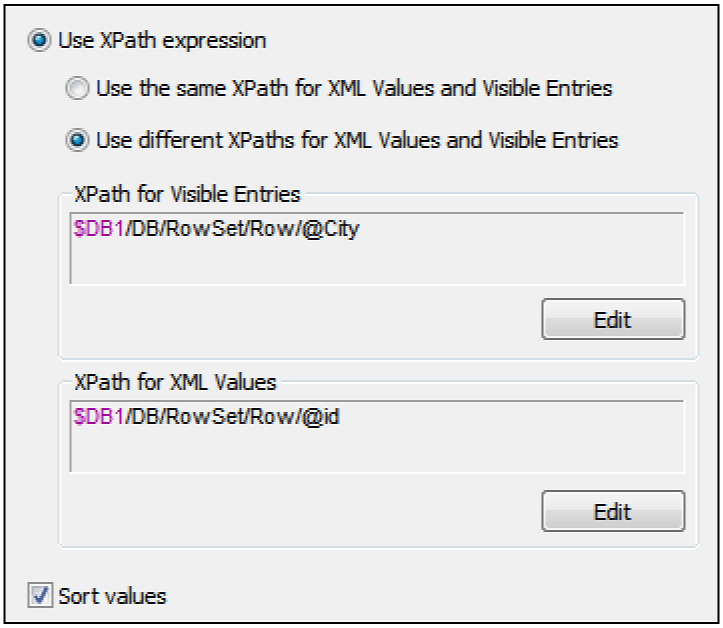
| Office | Year | Licenses |
|---|---|---|
| XPath: $DB1/DE | XPath: $XML1/r | XPath: sum($DE |

### The **DesiredOffice** combo box

The following settings have been made:

- A source node link is made between the combo box and the `$XML1/root/@DesiredOffice` node by dragging the node onto the combo box. This serves to pass the combo box selection to the node and the value of the node to the combo box.

- The items in the dropdown list of the combo box are defined in the Edit Combo Box dialog (*screenshot below*), which is accessed via the **Additional Dialog** button of the `Combo Box Entry Values` property (in the Styles & Properties Pane 274).



● Use XPath expression

○ Use the same XPath for XML Values and Visible Entries

● Use different XPaths for XML Values and Visible Entries

XPath for Visible Entries

$DB1/DB/RowSet/Row/@City

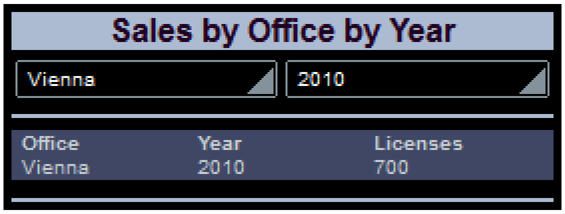
XPath for XML Values

$DB1/DB/RowSet/Row/@id

☑ Sort values

Notice that the values in the dropdown list are taken from the **`$DB1/DB/RowSet/Row/@City`** node (that is, the names of the cities). But the value that goes into the `$XML1/root/@DesiredOffice` node (because of the source node link) is taken from the **`$DB1/DB/RowSet/Row/@id`** node. Since the *Sort Values* check box has been selected the items of the dropdown list will be sorted.

- An *UpdateNode* action has been set for the **`OnFinishEditing`** event. Right-click the combo box and select **Control Actions for OnFinishEditing** to display the action definition. The node to be updated is **`root/@DesiredYear`**. The update value is provided by an XPath expression: `min(distinct-values($DB2/DB/RowSet/Row[@Office=$XML1/root/@DesiredOffice]/@Year))`. This expression selects all the records of the office selected in the combo box, then collects the unique years from these records, and finally selects the year with the minimum numerical value.



Sales by Office by Year

Vienna | 2010

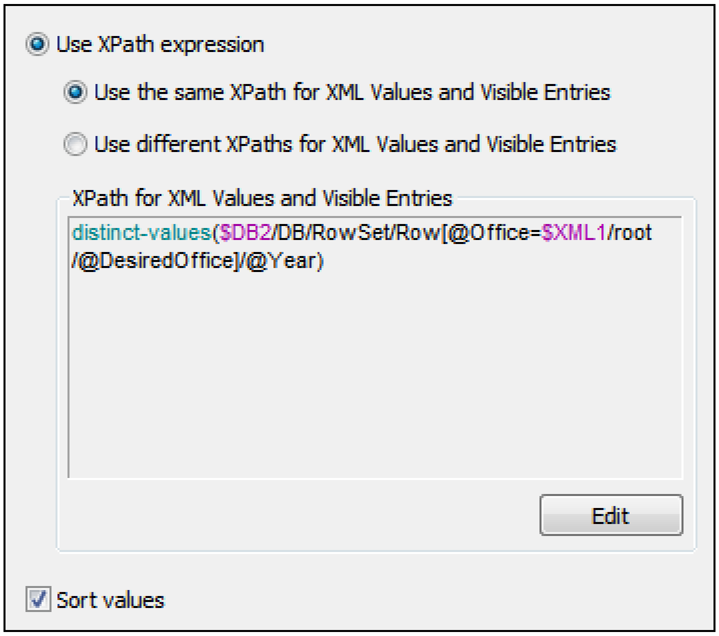
| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

So, when Vienna is selected in the first combo box (as in the screenshot above), all the records in **`$DB2`** with `@Office='Vienna'` were searched and a sequence of the unique years in these records was created. The year with the minimum numerical value—in this case **`2010`**—is passed to the node to be updated—in this case `$XML1/root/@DesiredOffice`. Since this node is the source node of the second combo box (the `@DesiredYear` combo box), this combo box now displays the minimum year value—in this case, **`2010`**.

## The **DesiredYear** combo box

The following settings have been made:

- A source node link is made between the combo box and the `$XML1/root/@DesiredYear` node by dragging the node onto the combo box. This serves to pass the combo box selection to the node and the value of the node to the combo box.
- The items in the dropdown list of the combo box are defined in the Edit Combo Box dialog (*screenshot below*), which is accessed via the **Additional Dialog** button of the `Combo Box Entry Values` property (in the Styles & Properties Pane 274).



● Use XPath expression

● Use the same XPath for XML Values and Visible Entries

○ Use different XPaths for XML Values and Visible Entries

XPath for XML Values and Visible Entries

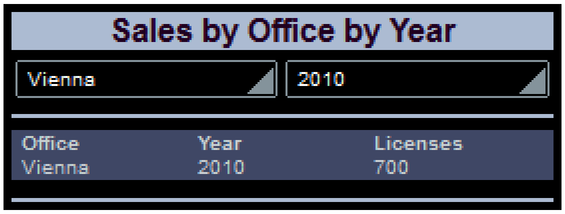
```
distinct-values($DB2/DB/RowSet/Row[@Office=$XML1/root
/@DesiredOffice]/@Year)
```

☑ Sort values

Notice that both the values in the dropdown list as well as those that will be passed to the XML node are the same. They are the sequence of all the unique years in which the selected office has recorded sales. Since the *Sort Values* check box has been selected the items of the dropdown list will be sorted.

## 4.5.5 The Tabular Report

The tabular report is displayed in the table below the two combo boxes. When the end user selects the office and year for which a report is required, the tabular report displays the total sales of that year (in terms of number of licenses). The screenshot below shows the page when the solution is run.

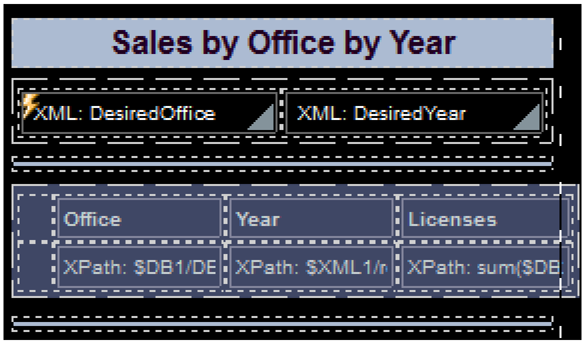


Sales by Office by Year

Vienna | 2010

| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

The next screenshot shows the tabular report in the design. The table consists of two rows and four columns, with the first column being used to provide padding. Each of the remaining six cells contains a label with a text value that is either directly entered text or is calculated by an XPath expression. See each label's `Text` property in the Styles & Properties Pane 274.



Sales by Office by Year

| Office | Year | Licenses |
|---|---|---|
| XPath: $DB1/DE | XPath: $XML1/r | XPath: sum($DB |

The XPath expressions are as follows:

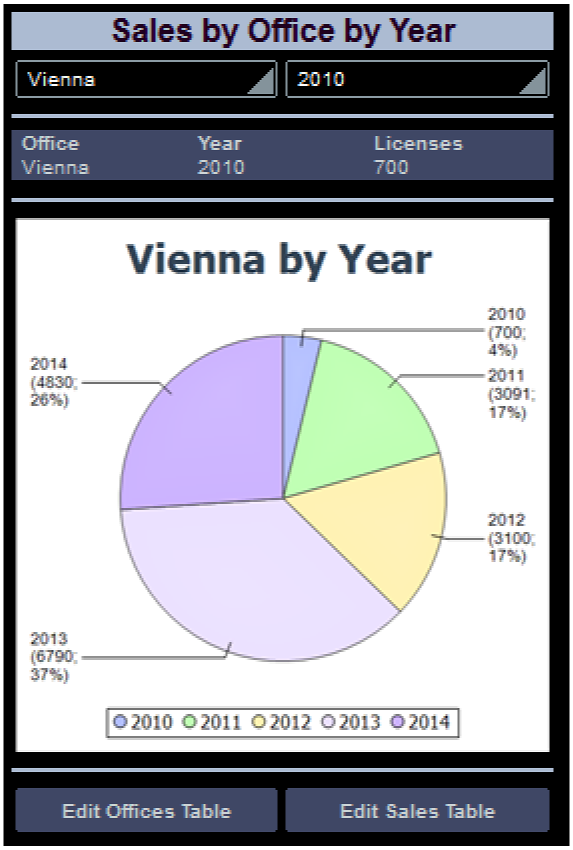
- *DesiredOffice:* Is taken from **$DB1**. It is the **@City** value of the **Row** with an **@id** equal to the **id** value of the combo box selection.
  **$DB1/DB/RowSet/Row[@id=$XML1/root/@DesiredOffice]/@City**

- *DesiredYear:* Is taken from **$XML1**. It is the value of the **@DesiredYear**. The year is either selected by the end user in the combo box, or is the minimum of all unique sales years for that office.
  **$XML1/root/@DesiredYear**

- *Licenses Sold:* Is taken from **$DB2**. Sums up all **@Licenses** values of the **Row** elements with **@Office** and **@Year** attributes equal to the values of the combo box selections. (Note that the **@Office** values in **$DB2** are the ID values of the offices, not their city names.)
  **sum($DB2/DB/RowSet/Row[@Office= $XML1/root/@DesiredOffice][@Year= $XML1/root/@DesiredYear]/@Licenses)**

## 4.5.6 The Charts

There are two charts in the design. The first chart shows the yearly breakdown of all sales of the office selected in the combo box (*see the simulator screenshot below*).



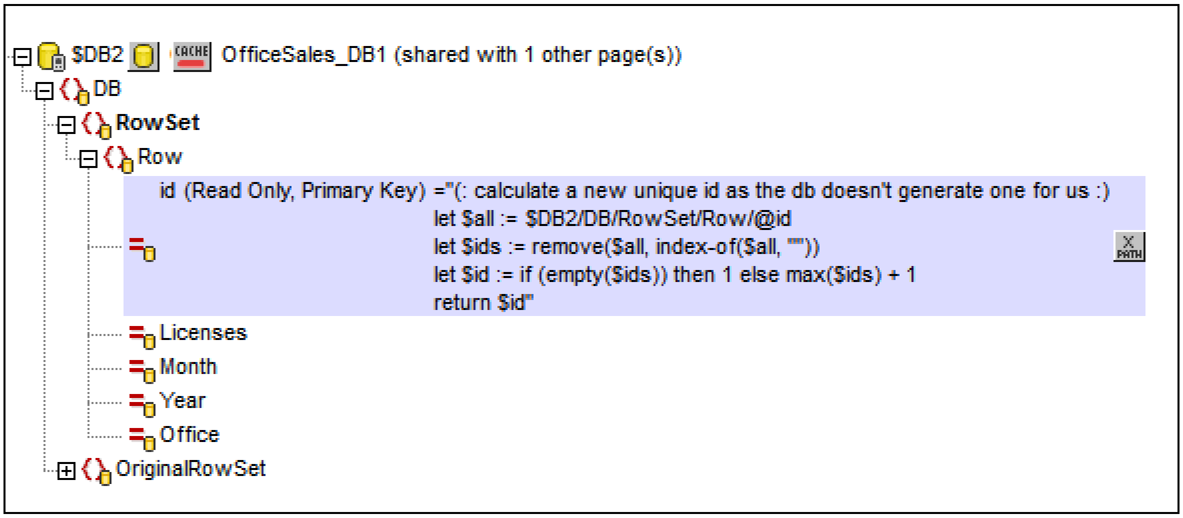
Sales by Office by Year

Vienna 2010

| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

## XPath context node

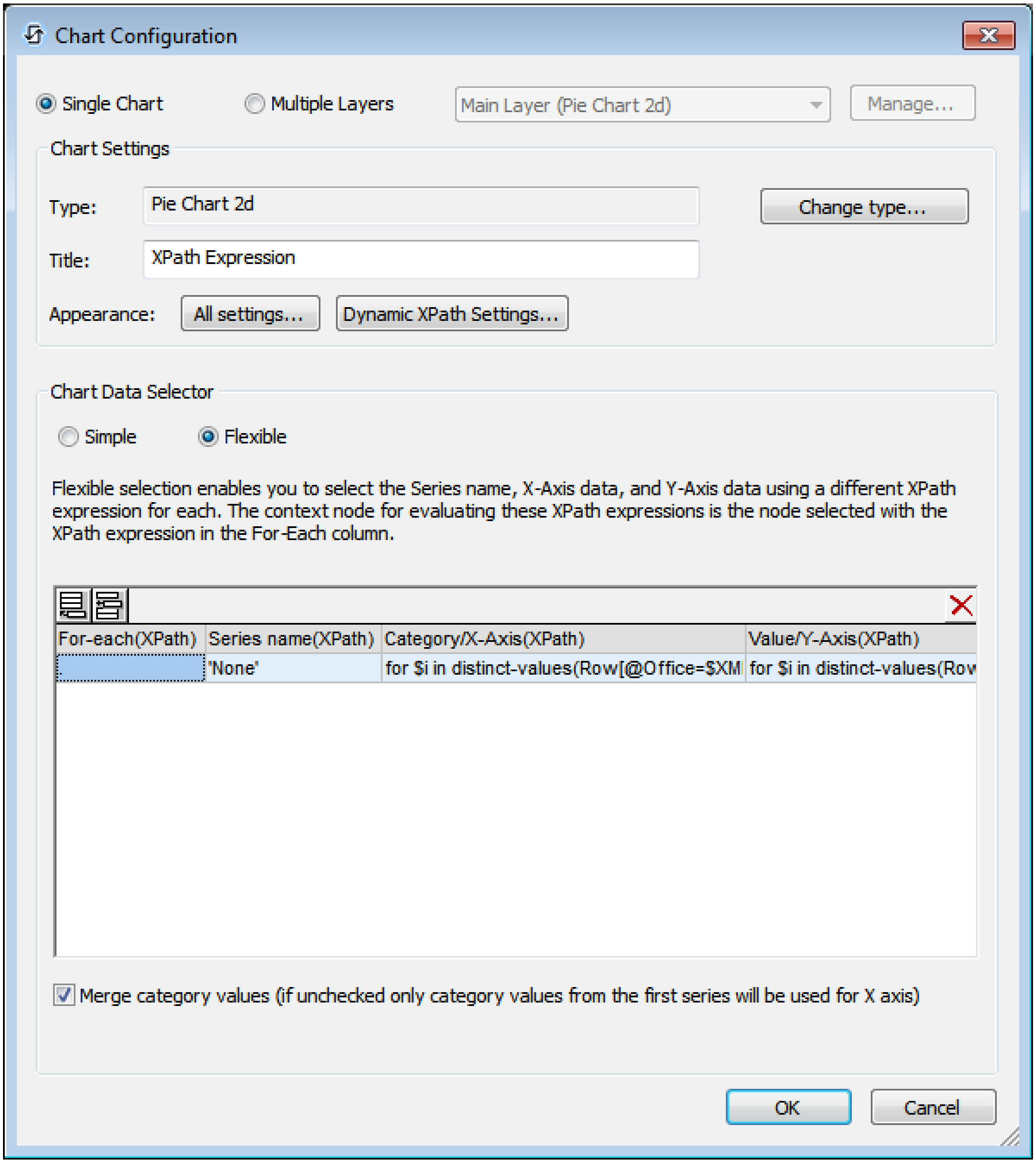
The main chart definitions are what goes on the X and Y axes. Since these are determined by XPath expressions it is important to correctly select the XPath context node for the chart. For the context node, it is best to select the immediate parent of the nodeset that will be used for the X and Y axes. Since we are going to use data from the Sales data table, we will use the **`$DB2`** tree for creating the chart (*screenshot below*). And since our nodeset for both axes will consist of the **`Row`** element, we select **`RowSet`** as the XPath context node. We do this by dragging the **`RowSet`** node onto the chart. The node is displayed bold, indicating that it is a source node.



```
$DB2  OfficeSales_DB1 (shared with 1 other page(s))
  DB
    RowSet
      Row
        id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                                    let $all := $DB2/DB/RowSet/Row/@id
                                    let $ids := remove($all, index-of($all, ""))
                                    let $id := if (empty($ids)) then 1 else max($ids) + 1
                                    return $id"
        Licenses
        Month
        Year
        Office
    OriginalRowSet
```

## Defining the chart axes

We are now ready to define the chart's axes. Open the chart's Chart Configuration dialog (*screenshot below*) by either double-clicking the chart or clicking the **Additional Dialog** button of the `Chart Settings` property (in the Styles & Properties Pane 274). Note that the chart type is a pie chart.



● Single Chart ○ Multiple Layers Main Layer (Pie Chart 2d)

Chart Settings

Type: Pie Chart 2d

Title: XPath Expression

Chart Data Selector

○ Simple ● Flexible

Flexible selection enables you to select the Series name, X-Axis data, and Y-Axis data using a different XPath expression for each. The context node for evaluating these XPath expressions is the node selected with the XPath expression in the For-Each column.

| For-each(XPath) | Series name(XPath) | Category/X-Axis(XPath) | Value/Y-Axis(XPath) |
|---|---|---|---|
| | 'None' | for $i in distinct-values(Row[@Office=$XM | for $i in distinct-values(Row |

☑ Merge category values (if unchecked only category values from the first series will be used for X axis)

For pie charts, we need two series (for the X and Y axes). The *Flexible* option is ideal for defining the axes for two series. The *For-each* setting selects the current node (`RowSet`). We define the following XPath expressions for the two axes:

- *X-Axis:* Creates a sequence of the unique years during which the selected office recorded sales.
  ```
  for $i in distinct-values(Row[@Office=$XML1/root/@DesiredOffice]/@Year) return $i
  ```

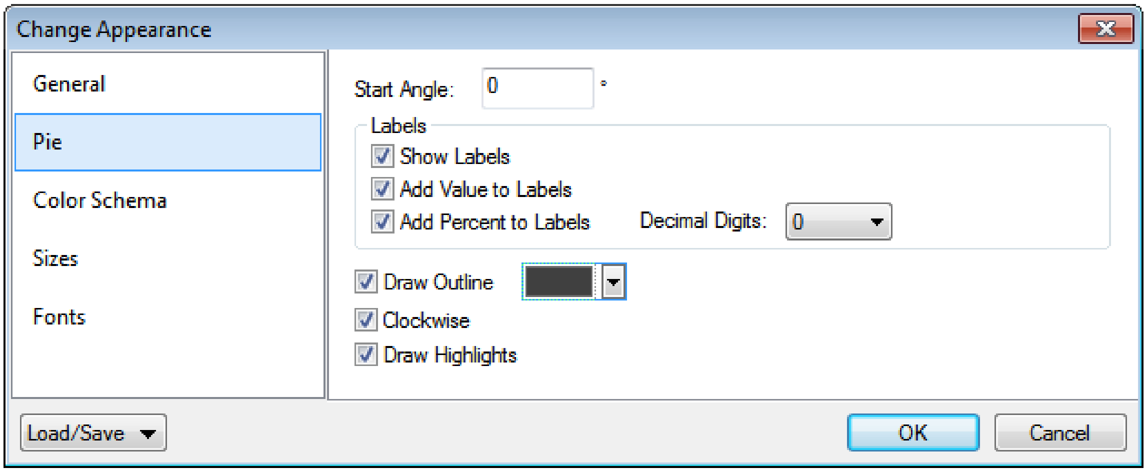
- *Y-Axis*: For the selected office and for each of its unique years, sums up the sales (stored in its `@Licenses` attribute)

```
for $i in distinct-values(Row[@Office=$XML1/root/@DesiredOffice]/@Year) return
sum(Row[@Office= $XML1/root/@DesiredOffice][@Year=$i]/@Licenses)
```

## Additional definitions

Additionally, the following settings were made:

- In the Chart Configuration dialog, click **Dynamic XPath Settings** and set the title using an XPath expression. This enables the selected office to be displayed in the title.
- In the Chart Configuration dialog, click **All Settings**. In the Change Appearance dialog that appears, select *Pie* and select *Add Value to Labels* and *Add Percent to Labels*.



Start Angle: 0 °

Labels

☑ Show Labels

☑ Add Value to Labels

☑ Add Percent to Labels Decimal Digits: 0

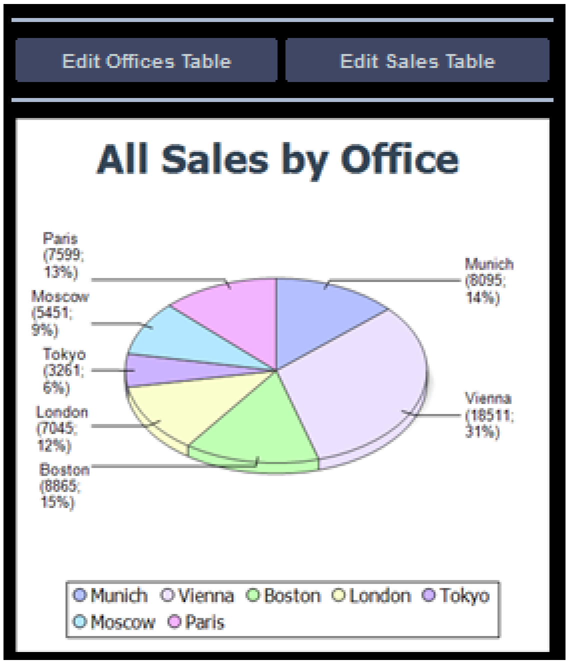
☑ Draw Outline

☑ Clockwise

☑ Draw Highlights

## The second chart

The second chart is similar to the first, but is a 3D pie chart (*screenshot below*). It shows the sales of each office over all years as a part of total sales over all years.



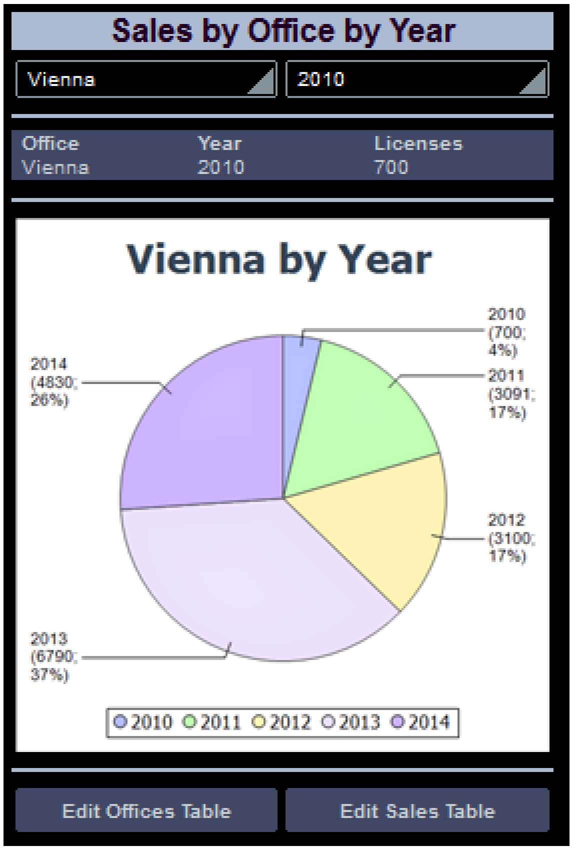
The XPath expressions are as follows:

- *X-Axis:* Creates a sequence of the city names of offices (not IDs), with city names being taken from **$DB1**.
  ```
  for $i in distinct-values(Row/@Office) return $DB1/DB/RowSet/Row[@id=$i]/@City
  ```

- *Y-Axis:* For the selected office, sums up the sales (stored in its `@Licenses` attribute)
  ```
  for $i in distinct-values(Row/@Office) return sum(Row[@Office=$i]/@Licenses)
  ```

## 4.5.7 Edit Offices Table

The Edit Offices table has been created on a separate top page. When the solution runs, this page is accessed from the main page (*screenshot below left*). Clicking the button **Edit Offices Table** loads the Edit Offices table (*screenshot below right*). The Offices table has seven rows, each of which has a non-editable ID column, an editable City column, and a Delete control (*see screenshot below right*). Additionally, there is an Append Row control below the last row, a **Submit** button in the *Edit Offices Table* bar, and a **Back** button to go back to the previous page (the main page in this case).

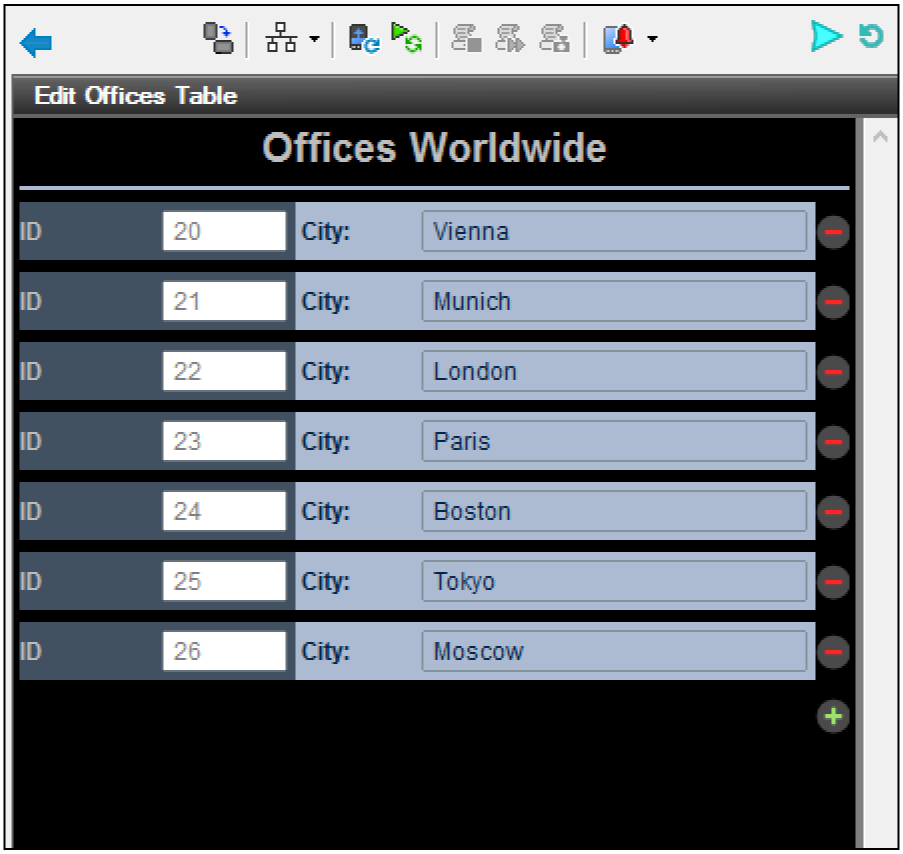


Sales by Office by Year

Vienna

2010

| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

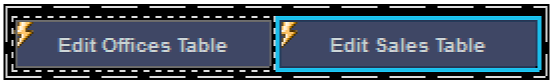


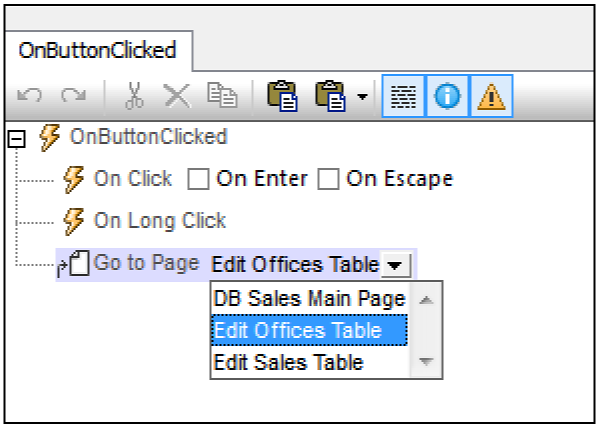
Edit Offices Table

Offices Worldwide

| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

In the design, the **Edit** buttons (*first screenshot below*) have both been assigned the *Go to Page* action on their respective **OnButtonClicked**[417] events (right-click the button and select **Control Actions for OnButtonClicked**). These *Go to Page* actions (*second screenshot below*) load the respective target pages.

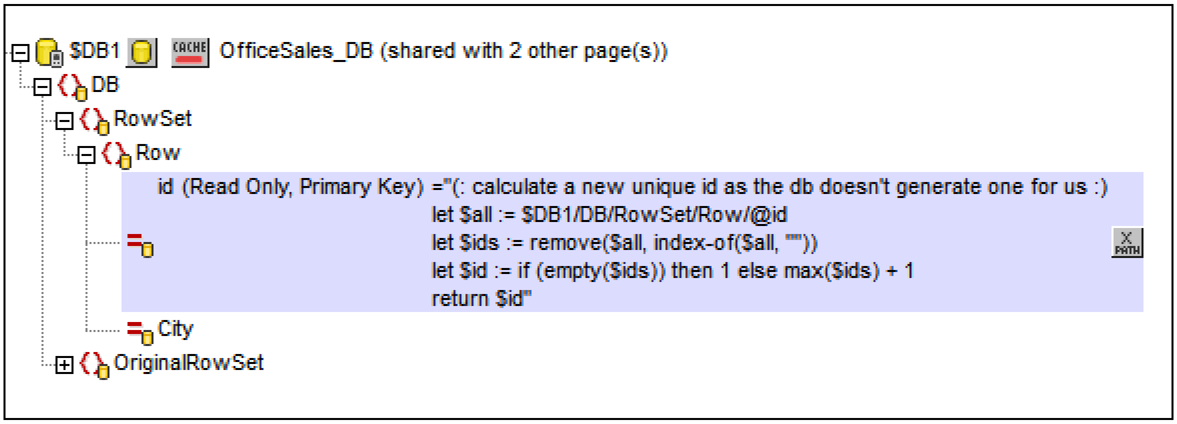




## Creating the editable Offices table

The Offices table in the DB has the structure displayed in the data tree of **$DB1** (*screenshot below*). Since the **@id** attribute is the primary key, it cannot be changed. This means that when a new record is appended, the end user cannot enter an `@id` value via the solution. The `@id` value must be generated automatically using an XQuery expression. The XQuery expression is inserted by using the context menu command, **Ensure Exists before Page Load (XPath Value)**:

```
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```



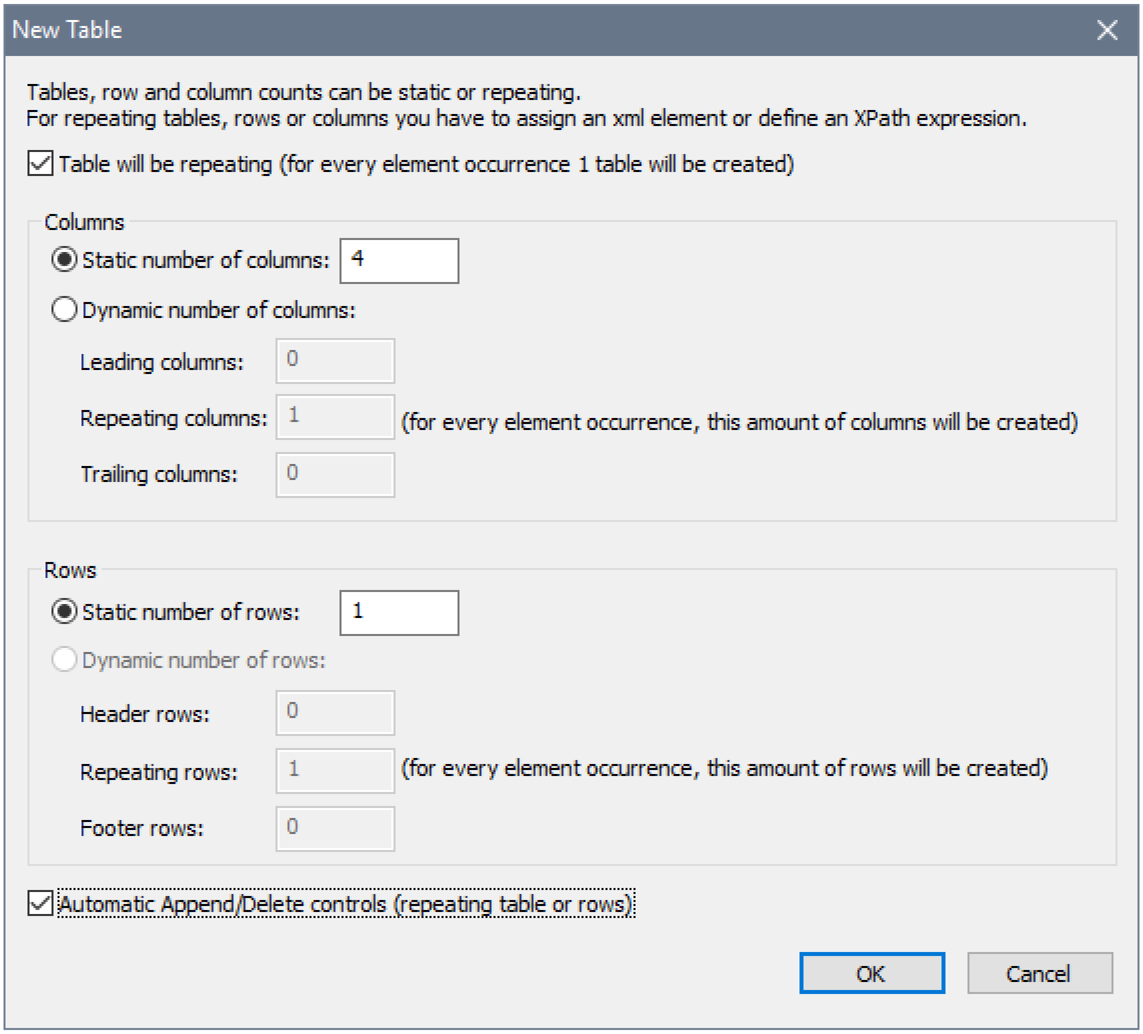
```
$DB1 OfficeSales_DB (shared with 2 other page(s))
  DB
    RowSet
      Row
        id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                                     let $all := $DB1/DB/RowSet/Row/@id
                                     let $ids := remove($all, index-of($all, ""))
                                     let $id := if (empty($ids)) then 1 else max($ids) + 1
                                     return $id"
        City
    OriginalRowSet
```

In the design, we will do the following:

| To... | How |
|---|---|
| Display all (`Office`) rows | Add a repeating table, with the `Office` row as the repeating element |
| Include controls for deletion and addition of rows | When adding the table, enable the automatic inclusion of Delete/Append controls |
| Enable editing of `@City` values | Add an Edit Field control that has a source node to `@City` |
| Save changes back to DB | Add a *Save* action to the page's `OnSubmitButtonClicked` event;<br>Also, right-click `$DB1` and toggle on **Create OriginalRowSet** |
| Go back to the main page | Add a *Go to Page* action to the page's `OnBackButtonClicked` event |

⊟ Add a repeating table that has Append/Delete controls

On dragging the table control from the Controls Pane[266] and dropping it in the design, the New table dialog appears (*screenshot below*).



New Table

Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

☑ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 4

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 1

○ Dynamic number of rows:

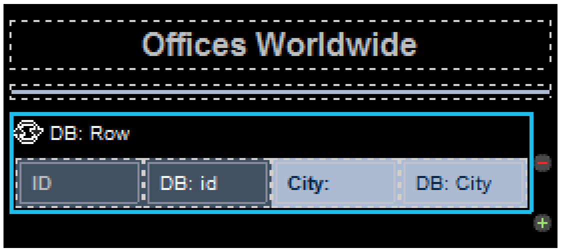
Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

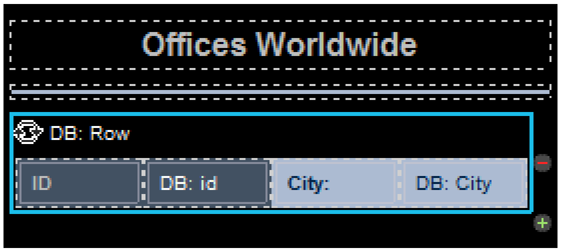
☑ Automatic Append/Delete controls (repeating table or rows)

Specify that the table will be repeating[1065], enter the number of columns (4) and rows (1), select the *Automatic Append/Delete Controls* check box, and click **OK**. Labels are added to the first three cells of the row, as shown in the screenshot below. A source node link to the `@id` node of `$DB1` is created for the second label (`DB:id`).



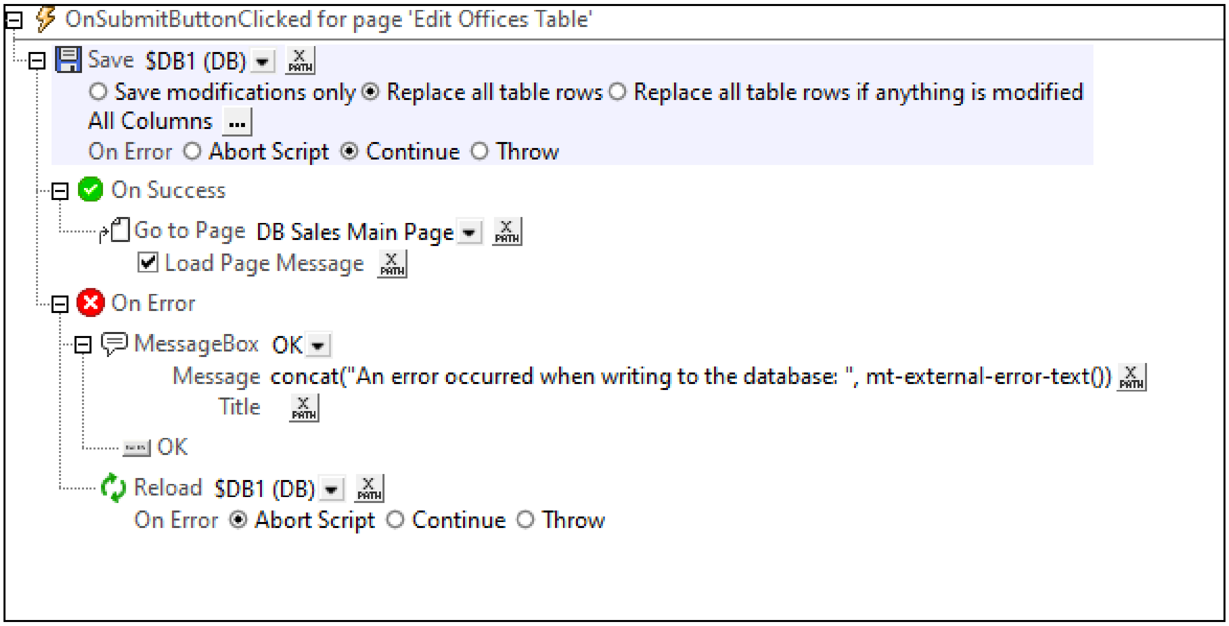
⊟_Enable editing of @City in $DB1

An edit field control is added to the fourth cell and a source node link to the `@City` node of `$DB1` is created for it (`DB:City`). We use an edit field control in this cell because we want the end user to be able edit `@City` values; all the other cells have label controls.



⊟ Page actions: 'Save' and 'Go to page'

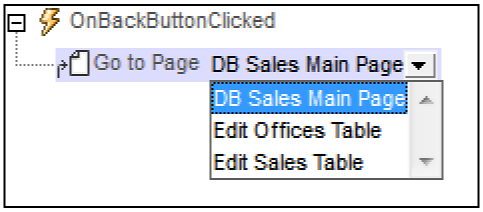
Click **Page | Page Actions** to open the Page Actions dialog (*screenshot below*).



```
⊟ OnSubmitButtonClicked for page 'Edit Offices Table'
  ⊟ Save $DB1 (DB)
      ○ Save modifications only ● Replace all table rows ○ Replace all table rows if anything is modified
      All Columns ...
      On Error ○ Abort Script ● Continue ○ Throw
    ⊟ On Success
        Go to Page DB Sales Main Page
          ☑ Load Page Message
    ⊟ On Error
      ⊟ MessageBox OK
            Message concat("An error occurred when writing to the database: ", mt-external-error-text())
              Title
          OK
        Reload $DB1 (DB)
          On Error ● Abort Script ○ Continue ○ Throw
```

Actions are defined for the following events:

- **`OnSubmitButtonClicked`**: Saves all columns of the page to the DB (`$DB1`) and goes back to the main page. You might also want to add the Reload action so that the DB is reloaded with the unmodified data in case the record is not saved to the DB (*see screenshot above*).
- **`OnBackButtonClicked`**: Goes back to the main page.

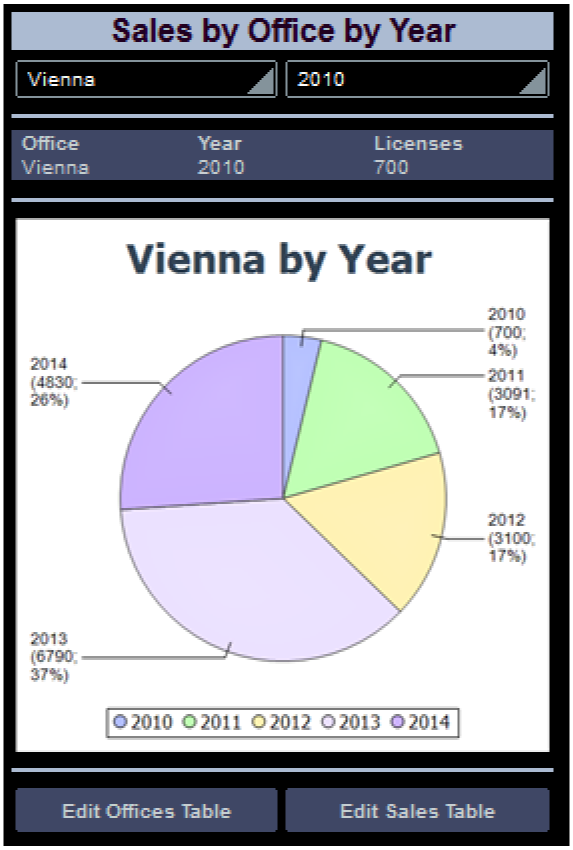


```
⊟ OnBackButtonClicked
    Go to Page DB Sales Main Page
                 DB Sales Main Page
                 Edit Offices Table
                 Edit Sales Table
```

The tree of the page source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. Original data is saved in the `OriginalRowSet` element, so that the columns of the `RowSet` element can be edited. The `OriginalRowSet` element is updated with the new value only when the data is saved back to the DB.

## 4.5.8 Edit Sales Table

The Edit Sales table, like the Edit Offices Table, has been created on a separate top page. When the solution runs, this page is accessed from the main page (*screenshot below left*). Clicking the button **Edit Sales Table** loads the Edit Sales table (*screenshot below right*). The Sales table has multiple rows, each of which has a non-editable (sales item) ID column, editable Office, Month, Year, and Licenses columns, and a Delete control (*see screenshot below right*). Additionally, there is an Append Row control below the last row, a **Submit** button in the *Edit Sales Table* bar, and a **Back** button to go back to the previous page (the main page in this case).

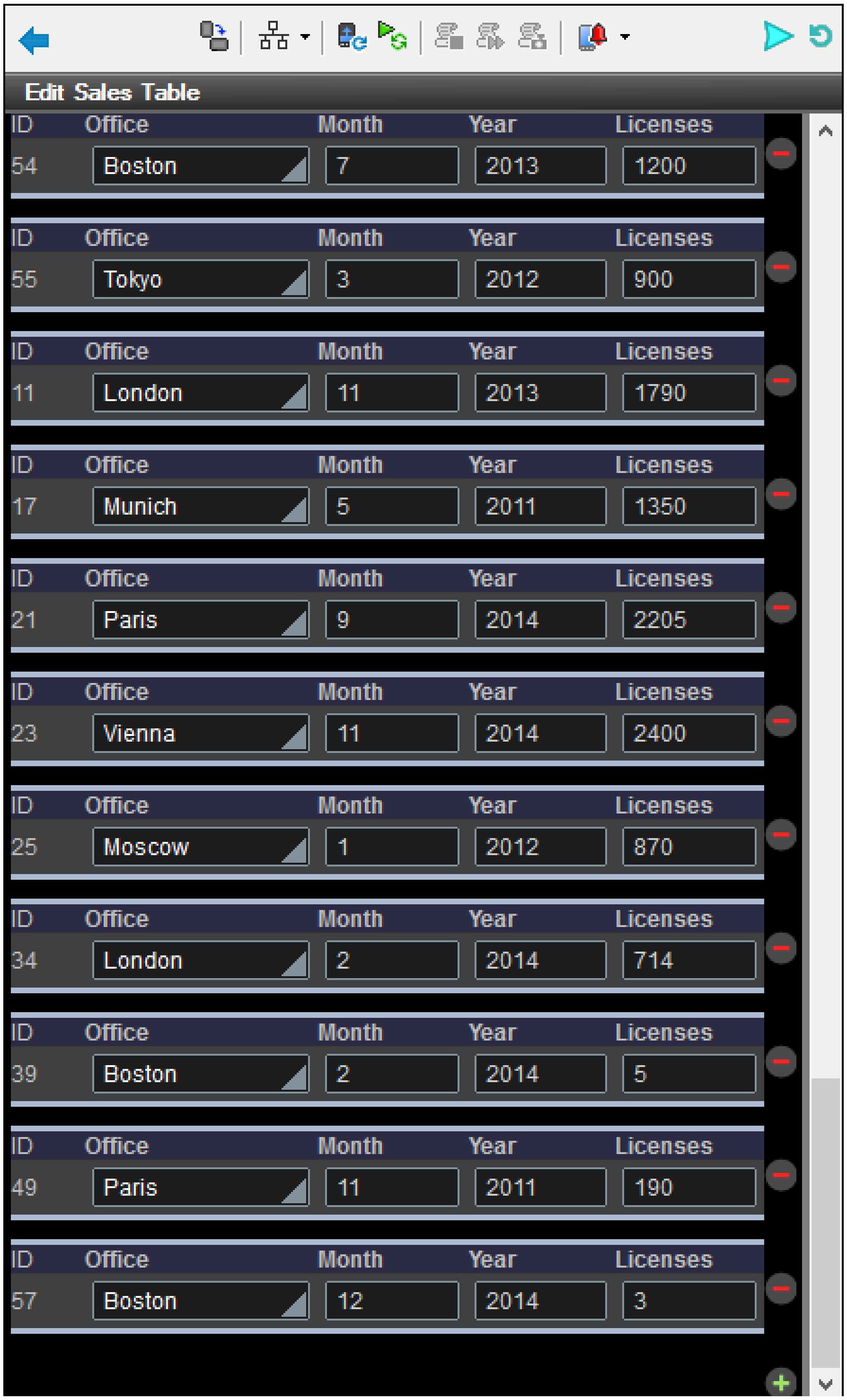


Sales by Office by Year

Vienna

2010

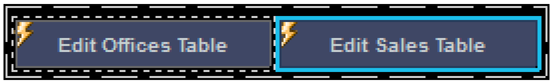
| Office | Year | Licenses |
|---|---|---|
| Vienna | 2010 | 700 |

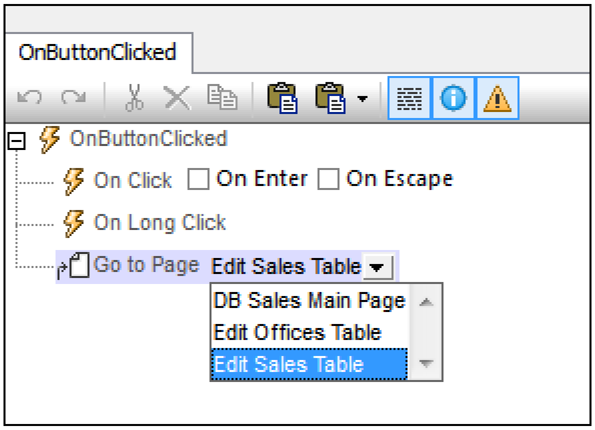


Edit Sales Table

| ID | Office | Month | Year | Licenses |
|---|---|---|---|---|
| 54 | Boston | 7 | 2013 | 1200 |
| 55 | Tokyo | 3 | 2012 | 900 |
| 11 | London | 11 | 2013 | 1790 |
| 17 | Munich | 5 | 2011 | 1350 |
| 21 | Paris | 9 | 2014 | 2205 |
| 23 | Vienna | 11 | 2014 | 2400 |
| 25 | Moscow | 1 | 2012 | 870 |
| 34 | London | 2 | 2014 | 714 |
| 39 | Boston | 2 | 2014 | 5 |
| 49 | Paris | 11 | 2011 | 190 |
| 57 | Boston | 12 | 2014 | 3 |

In the design, the **Edit** buttons (*first screenshot below*) have both been assigned the *Go to Page* action on their respective `OnButtonClicked`[417] events (right-click the button and select **Control Actions for OnButtonClicked**). These *Go to Page* actions (*second screenshot below*) load the respective target pages.

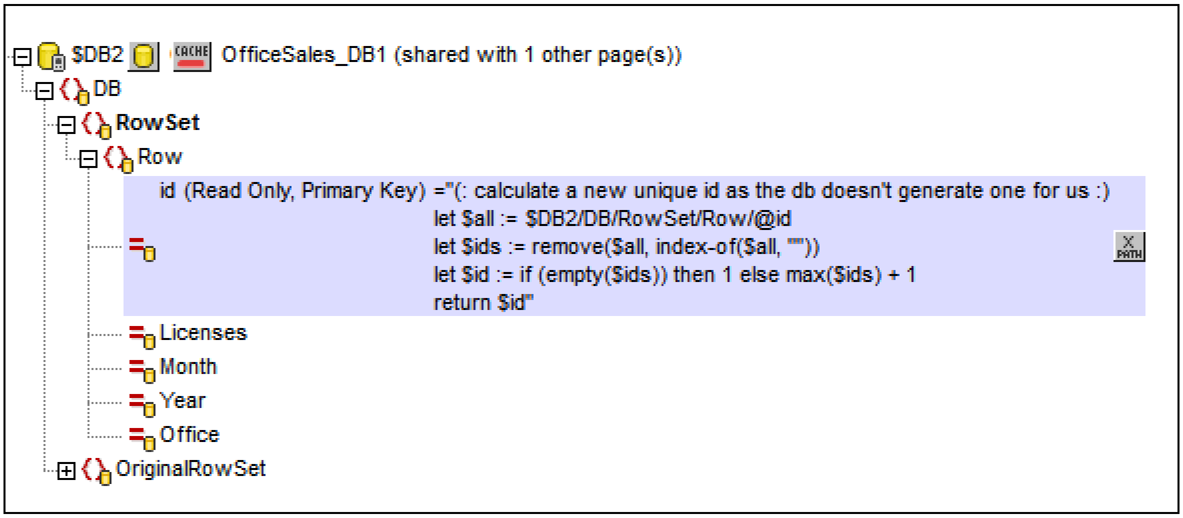




## Creating the editable Sales table

The Sales table in the DB has the structure displayed in the data tree of **`$DB2`** (*screenshot below*). Since the **`@id`** attribute is the primary key, it cannot be changed. This means that when a new record is appended, the end user cannot enter an `@id` value via the solution. The `@id` value must be generated automatically using an XQuery expression. The XQuery expression is inserted by using the context menu command, **Ensure Exists before Page Load (XPath Value)**:

```
let $all := $DB2/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```



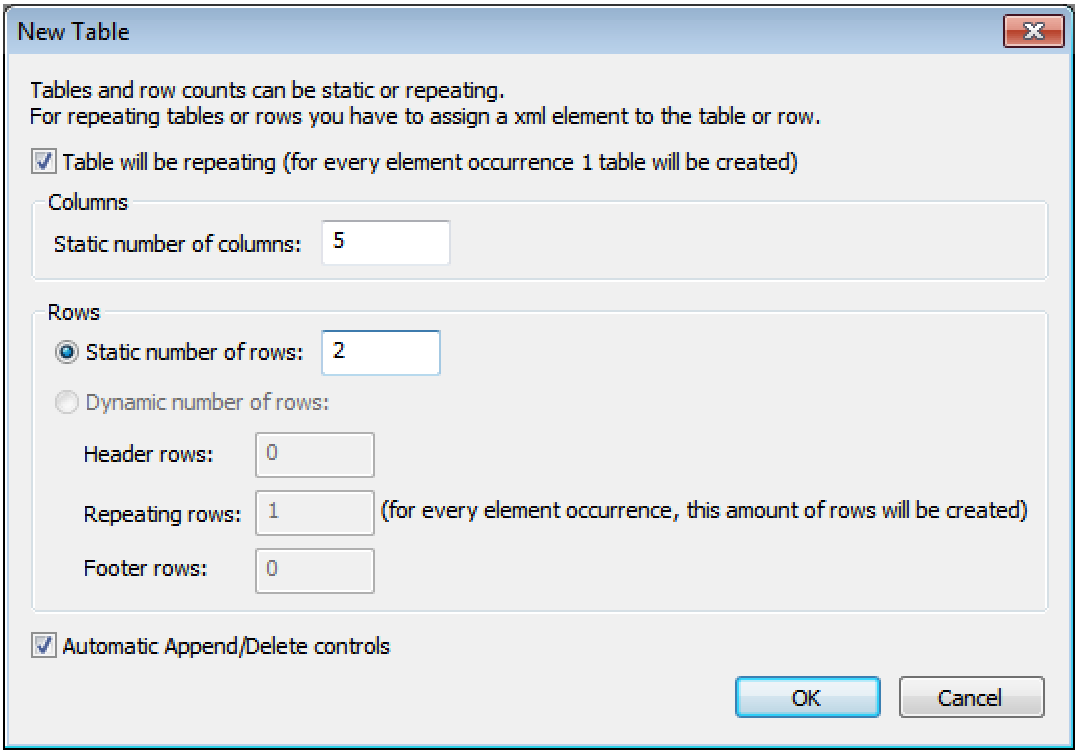
```
$DB2 OfficeSales_DB1 (shared with 1 other page(s))
  DB
    RowSet
      Row
        id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                                    let $all := $DB2/DB/RowSet/Row/@id
                                    let $ids := remove($all, index-of($all, ""))
                                    let $id := if (empty($ids)) then 1 else max($ids) + 1
                                    return $id"
        Licenses
        Month
        Year
        Office
    OriginalRowSet
```

In the design, we will do the following:

| To... | Do this... |
|---|---|
| Display all (`Sales`) rows | Add a repeating table, with the `Sales` row as the repeating element |
| Include controls for deletion and addition of rows | When adding the table, enable the automatic inclusion of Delete/Append controls |
| Enable editing of editable values | Add a combo box and edit field controls that have page source links |
| Save changes back to DB | Add a *Save* action to the page's `OnSubmitButtonClicked` event; Also, right-click `$DB2` and toggle on **Create OriginalRowSet** |
| Go back to the main page | Add a *Go to Page* action to the page's `OnBackButtonClicked` event |

⊟ Add a repeating table that has Append/Delete controls

On dragging the table control from the Controls Pane[266] and dropping it in the design, the New table dialog appears (*screenshot below*).



Tables and row counts can be static or repeating.
For repeating tables or rows you have to assign a xml element to the table or row.

☑ Table will be repeating (for every element occurrence 1 table will be created)

Columns

Static number of columns: 5

Rows

● Static number of rows: 2

○ Dynamic number of rows:

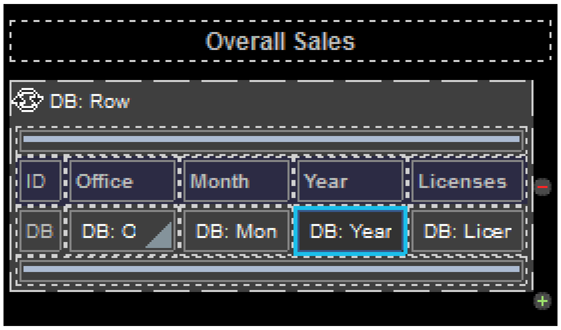
Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

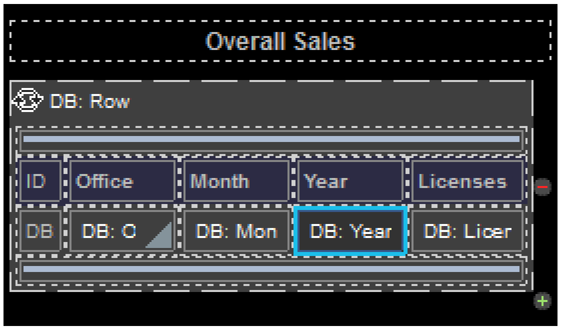
☑ Automatic Append/Delete controls

Specify that the table will be repeating[1065], enter the number of columns `(5)` and rows `(2)`, select the *Automatic Append/Delete Controls* check box, and click **OK**. Labels are added for headers to the cells of the first row. A label is added for the uneditable `@id` value to the first cell of the second row. A source node link to the `@id` node of `$DB2` is created on this label (`DB:id`).



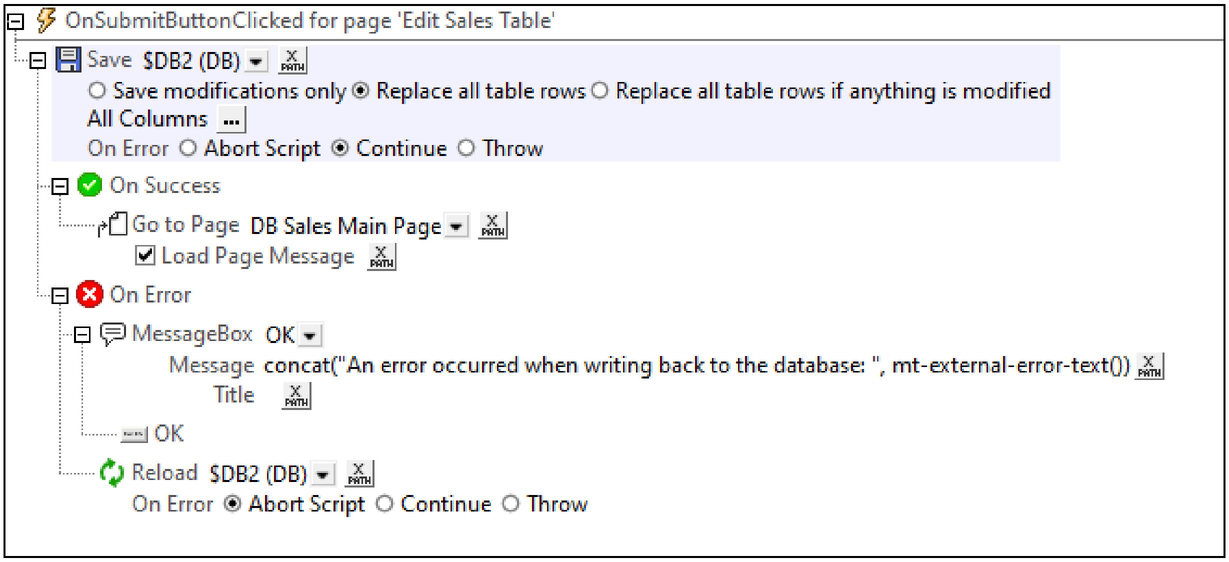
⊟ Enable editing of the editable nodes

A combo box is added for the office (with a source node link to `@Office`), and edit fields are added for the month, year, and licenses cells, with page source links to the respective nodes.



⊟ Page actions: 'Save' and 'Go to page'

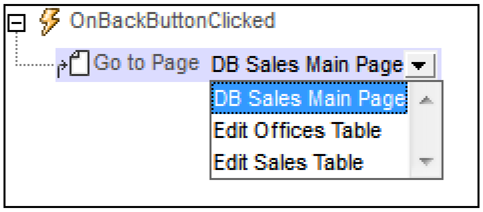
Click **Page | Page Actions** to open the Page Actions dialog (*screenshot below*).



```
⊟ OnSubmitButtonClicked for page 'Edit Sales Table'
  ⊟ Save  $DB2 (DB)
      ○ Save modifications only ● Replace all table rows ○ Replace all table rows if anything is modified
      All Columns ...
      On Error ○ Abort Script ● Continue ○ Throw
    ⊟ On Success
        Go to Page  DB Sales Main Page
          ☑ Load Page Message
    ⊟ On Error
      ⊟ MessageBox  OK
            Message concat("An error occurred when writing back to the database: ", mt-external-error-text())
               Title
          OK
        Reload  $DB2 (DB)
          On Error ● Abort Script ○ Continue ○ Throw
```

Actions are defined for the following events:

- **`OnSubmitButtonClicked`**: Saves all columns of the page to the DB (`$DB1`) and goes back to the main page. You might also want to add the Reload action so that the DB is reloaded with the unmodified data in case the record is not saved to the DB (*see screenshot above*).
- **`OnBackButtonClicked`**: Goes back to the main page.

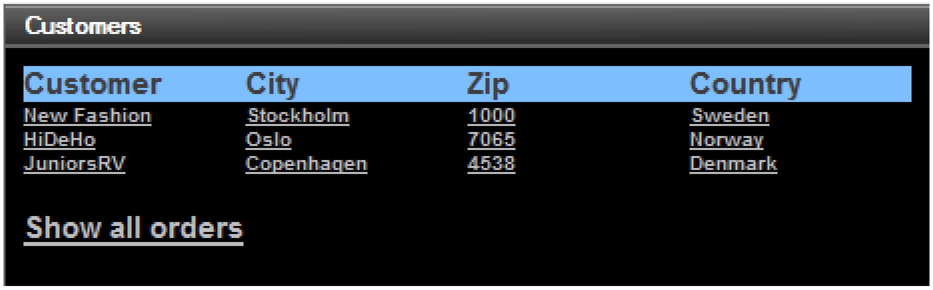


```
⊟ OnBackButtonClicked
    Go to Page  DB Sales Main Page
                DB Sales Main Page
                Edit Offices Table
                Edit Sales Table
```

The tree of the page source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. Original data is saved in the `OriginalRowSet` element, so that the columns of the `RowSet` element can be edited. The `OriginalRowSet` element is updated with the new value only when the data is saved back to the DB.

## 4.6 SubPages-And-Visibility

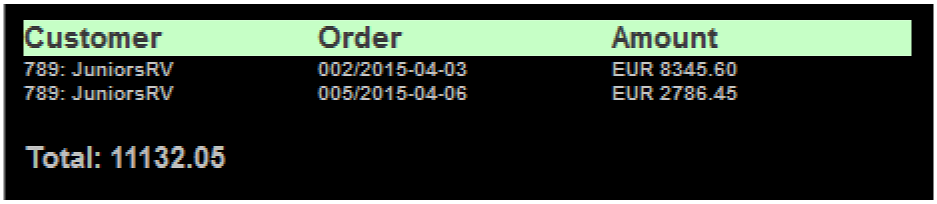
This tutorial shows you how to open a sub page from a top page, and how to filter the display of a data structure using the `Visible` property. It also describes how to use dynamic tables, action groups, the Update Node action, and decimals in XPath functions. The top page (*first screenshot below*) displays all the customers that are currently stored in the database. If the end user clicks a customer detail (name, city, etc), a sub page opens showing the current orders of that customer (*second screenshot below*).



Customers

| Customer | City | Zip | Country |
|---|---|---|---|
| New Fashion | Stockholm | 1000 | Sweden |
| HiDeHo | Oslo | 7065 | Norway |
| JuniorsRV | Copenhagen | 4538 | Denmark |

Show all orders



| Customer | Order | Amount |
|---|---|---|
| 789: JuniorsRV | 002/2015-04-03 | EUR 8345.60 |
| 789: JuniorsRV | 005/2015-04-06 | EUR 2786.45 |

Total: 11132.05

The data for the design is stored in two page sources: one that stores customer data, and a second that stores order details. The two page sources have a customer code column in common, which is used to connect customer data with order details. We use XML files in this tutorial, but the page sources could as easily be databases, in which the customer code column is used as the primary key.

### The tutorial files

The files for this tutorial are located in your ([72]My) Documents[72] MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\SubPagesAndVisibility`**.

- The XML data file that contains customer data is `Customers.xml`
- The XML data file that contains order data is `Orders.xml`
- The design file you will end with should be similar to `SubPagesAndVisibility.mtd`

### Tutorial structure

This tutorial is organized into the following sections:

- Design Structure[189]
- Page source Listings[190]
- Top Page: Page Sources[192]
- Top Page: Customers Table[194]
- Top Page: Action Group, Go to Sub Page[196]
- Top Page: Show All Orders Action[198]
- Sub Page: Page Sources[199]
- Sub Page: Orders Table[200]

- Sub Page: Visibility Property[202]
- Sub Page: Decimal Totals in XPath[203]
- Simulation and Testing[204]

## 4.6.1 Design Structure

The design file contains a top page (named Customers) and a sub page (named Orders). The top page (Customers, *first screenshot below*) displays all the customers that are currently stored in the database. If the end user clicks a customer detail on the top page, a sub page (the Orders page, *second screenshot below*) opens. This page shows the current orders of that customer. The top page also has an option to display, on the sub page, all current orders in the database, that is, the orders of all customers.

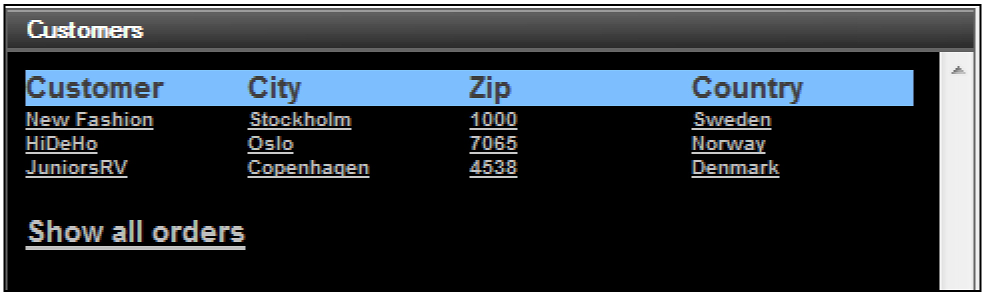
The key mechanism of this design is that of displaying (in the sub page) the orders of only the one customer that is selected on the top page. This is achieved with the **`Visible`** property of a table that displays all the current orders. The property specifies which elements should be visible, and so acts as a display filter. We will specify, via an XPath expression, that only the orders of the selected customer will be visible. This filtered table is a simple and effective alternative to creating a customer-specific table that contains only the orders of the selected customer.

### Design steps

The design will be built up as described below. (*The screenshots show simulations of the completed design.*)

*Top page: Customers*

- Create the top page and two page sources[192]: **`$XML1`** and **`$CUSTOMERS`**
- Create a dynamic table to contain customer data[194] from `$CUSTOMERS`. Each row of the table will correspond to one customer in the XML page source `$CUSTOMERS`
- Create a sub page named Orders[196]
- Create an action group[196] that does the following: (i) update nodes in `$XML1` with data about the customer node that the user clicks; (ii) goes to the sub page named Orders
- Assign the action group to each label[196] that contains customer data. As a result, when some customer data is clicked, then the action group is executed
- Create a label to show all orders[198]. The action of this label (show all orders) is in contrast to the other Go to Sub Page actions, which show the orders of a single selected customer



Customers

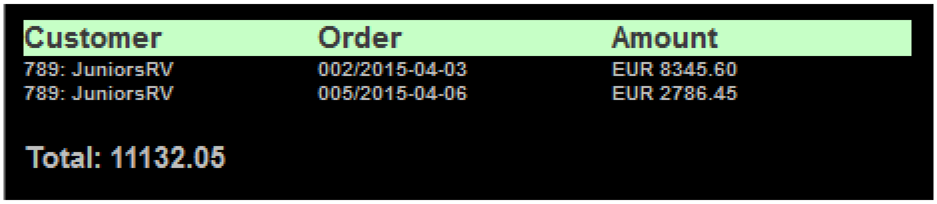
| Customer | City | Zip | Country |
|---|---|---|---|
| New Fashion | Stockholm | 1000 | Sweden |
| HiDeHo | Oslo | 7065 | Norway |
| JuniorsRV | Copenhagen | 4538 | Denmark |

Show all orders

*Sub page: Orders*

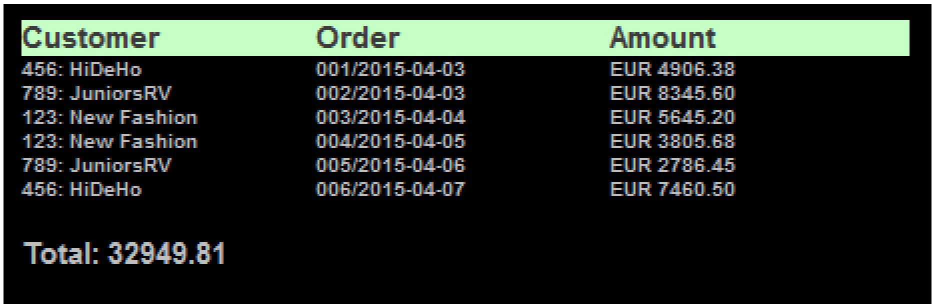
- Create the three page sources for the sub page[199]: **`$XML1`** (shared with top page), **`$CUSTOMERS`** (shared with top page), and **`$ORDERS`**

- Create a dynamic table to display the order details in the data file[200] (*screenshots below show the order tables of (i) a selected customer, and (ii) all customers*). Each row of the table will correspond to one order in the XML page source `$ORDERS`
- Set up the visibility property of the table's repeating row group[202] to show (i) only the customer selected on the top page, or (ii) all customers
- Create an XPath expression to generate the total amount[203] of (i) all orders of the selected customer, or (ii) all current orders



| Customer | Order | Amount |
|---|---|---|
| 789: JuniorsRV | 002/2015-04-03 | EUR 8345.60 |
| 789: JuniorsRV | 005/2015-04-06 | EUR 2786.45 |

Total: 11132.05



| Customer | Order | Amount |
|---|---|---|
| 456: HiDeHo | 001/2015-04-03 | EUR 4906.38 |
| 789: JuniorsRV | 002/2015-04-03 | EUR 8345.60 |
| 123: New Fashion | 003/2015-04-04 | EUR 5645.20 |
| 123: New Fashion | 004/2015-04-05 | EUR 3805.68 |
| 789: JuniorsRV | 005/2015-04-06 | EUR 2786.45 |
| 456: HiDeHo | 006/2015-04-07 | EUR 7460.50 |

Total: 32949.81

## 4.6.2 Page Source Listings

The design will have three XML page sources:

- **`$XML1`** is a tree that is created directly in the design. Its purpose is to store the end-user's selection of which customer's order details to display in the sub page. The `$XML1` page source is shared by both pages of the design.
- **`$CUSTOMERS`** contains the details of three customers. The XML tree structure and the customer data are imported from the XML file `Customers.xml` (*see listing below*[190]). The `$CUSTOMERS` page source is used in both the top page as well as the sub page.
- **`$ORDERS`** contains the details of six orders placed by the three customers of `Customers.xml`. The XML tree structure and the order details are imported from the XML file `Orders.xml` (*see listing below*[191]). The orders in the `$ORDERS` page source are displayed in a table in the Orders sub page.

⊟ *Listing of the XML page source, Customers.xml*

Located in the MobileTogether folder of the *([72]My) Documents folder*[72]: **`MobileTogetherDesignerExamples\Tutorials\Customers.xml`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<Customers>
  <Customer code="123">
    <Name>New Fashion</Name>
    <AddressLine01>56 Tromer Street</AddressLine01>
    <AddressLine02></AddressLine02>
    <City>Stockholm</City>
    <ZipCode>1000</ZipCode>
    <Country>Sweden</Country>
    <Email>contact01@newfashion.dummy</Email>
    <Phone/>
  </Customer>
  <Customer code="456">
    <Name>HiDeHo</Name>
    <AddressLine01>7 Norsk Street</AddressLine01>
    <AddressLine02></AddressLine02>
    <City>Oslo</City>
    <ZipCode>7065</ZipCode>
    <Country>Norway</Country>
    <Email>contact02@hideho.dummy</Email>
    <Phone/>
  </Customer>
  <Customer code="789">
    <Name>JuniorsRV</Name>
    <AddressLine01>81 Bjork Street</AddressLine01>
    <AddressLine02></AddressLine02>
    <City>Copenhagen</City>
    <ZipCode>4538</ZipCode>
    <Country>Denmark</Country>
    <Email>contact03@juniorsrus.dummy</Email>
    <Phone/>
  </Customer>
</Customers>
```

⊟ *Listing of the XML page source, Orders.xml*

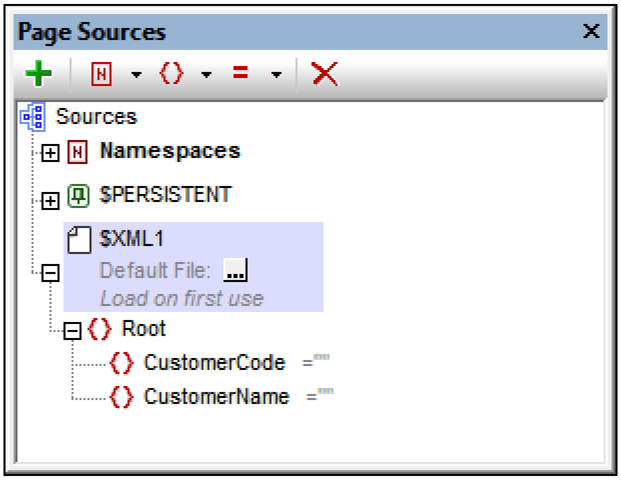
Located in the MobileTogether folder of the ([72]My) Documents folder[72]: **MobileTogetherDesignerExamples\Tutorials\AltovaProducts.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<Orders>
  <Order number="001">
    <CustomerCode>456</CustomerCode>
    <OrderDate>2015-04-03</OrderDate>
    <OrderAmount>4906.38</OrderAmount>
    <Currency>EUR</Currency>
  </Order>
  <Order number="002">
    <CustomerCode>789</CustomerCode>
    <OrderDate>2015-04-03</OrderDate>
    <OrderAmount>8345.60</OrderAmount>
    <Currency>EUR</Currency>
  </Order>
  <Order number="003">
    <CustomerCode>123</CustomerCode>
    <OrderDate>2015-04-04</OrderDate>
    <OrderAmount>5645.20</OrderAmount>
    <Currency>EUR</Currency>
  </Order>
  <Order number="004">
    <CustomerCode>123</CustomerCode>
    <OrderDate>2015-04-05</OrderDate>
    <OrderAmount>3805.68</OrderAmount>
    <Currency>EUR</Currency>
  </Order>
  <Order number="005">
    <CustomerCode>789</CustomerCode>
    <OrderDate>2015-04-06</OrderDate>
    <OrderAmount>2786.45</OrderAmount>
    <Currency>EUR</Currency>
  </Order>
  <Order number="006">
    <CustomerCode>456</CustomerCode>
    <OrderDate>2015-04-07</OrderDate>
    <OrderAmount>7460.50</OrderAmount>
    <Currency>EUR</Currency>
  </Order>
</Orders>
```

## 4.6.3 Top Page: Page Sources

In this section, we will create the top page and its page sources. Do this as follows:

1. Create a new design file with the **File | New** command.
2. Save the file with any name you like.
3. In the Pages Pane[257], rename the top page (which is created by default) to `Customers`. Do this by double-clicking the page name and then editing it.
4. In the Page Sources Pane[270], add a new empty XML source by clicking the **Add Source**[270] icon, and selecting *New, empty XML*[317]. In the second screen, keep the default selections. A page source called `$XML1` will be created (*see screenshot below*).
5. Manually build the structure of this page source so that it is as shown in the screenshot below. Do this by right-clicking nodes in the tree (starting with `$XML1`) and using the **Add Child**, **Append**, and **Insert** context menu commands.
6. Right-click the **`CustomerCode`** node, and select the command **Ensure Exists on Load (fixed value)**[371]. In the value box that appears for the node, press **Enter** without having entered any value. Do the same for the **`CustomerName`** node. This ensures that the two nodes are created empty in the `$XML1` tree when the page loads.

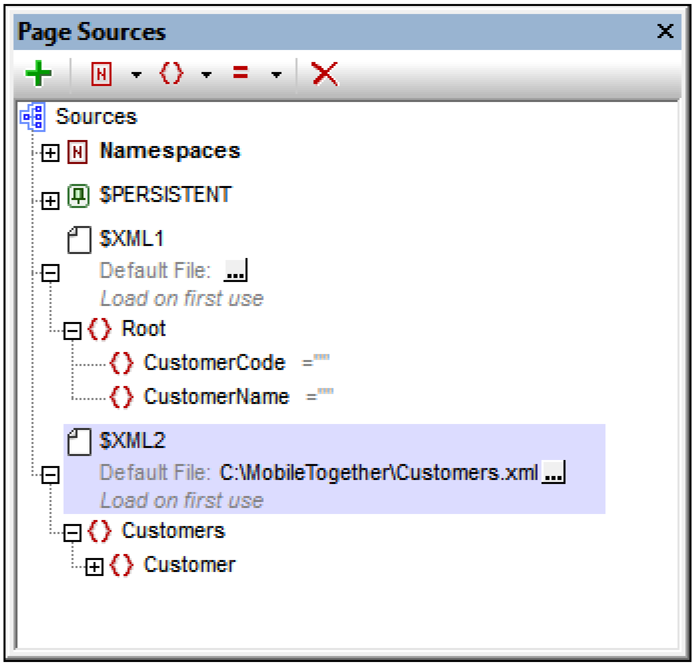


Page Sources

Sources
Namespaces
$PERSISTENT
$XML1
Default File:
Load on first use
Root
CustomerCode =""
CustomerName =""

At runtime, we plan to have the nodes of the `$XML1` tree updated with the data (code and name) of the customer that the end user selects from the list of customers displayed on this page.

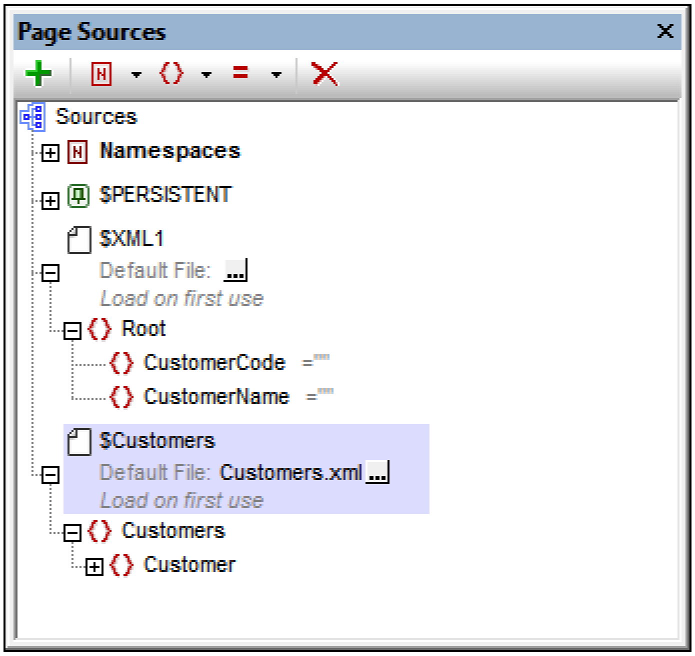
7. Create a second page source by clicking the **Add Source** 270 icon, and selecting *New XML or HTML structure imported from file* 317. Browse for the file `Customers.xml` 190 and click **Open**. When you are prompted about whether to deploy the file, choose **Yes**. A page source called `$XML2` will be created (*see screenshot below*).



Page Sources

Sources
Namespaces
$PERSISTENT
$XML1
Default File:
Load on first use
Root
CustomerCode =""
CustomerName =""
$XML2
Default File: C:\MobileTogether\Customers.xml
Load on first use
Customers
Customer

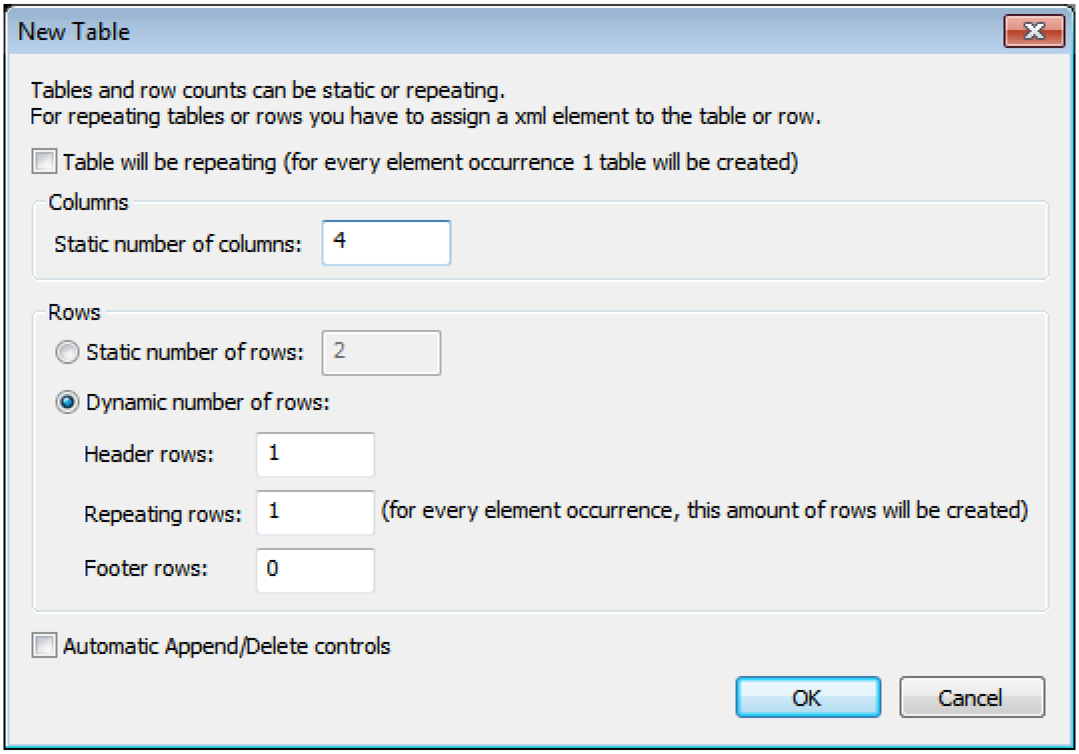
8. Double-click the root node `$XML2` and edit the name to `$CUSTOMERS` (*see screenshot below*).
9. Click the **Additional Dialog** button of the `$CUSTOMERS` default file 355. In the dialog that appears, select the *Relative paths* check box to make the file's path relative to the design (*see screenshot below*).



## 4.6.4 Top Page: Customers Table

We will now create a table to display details of all the customers currently stored in the XML page source `Customers.xml` 190. Create the table as follows.

1. Drag a Table control 614 from the Controls Pane 266 and drop it into the design.
2. In the New Table dialog that appears (*see screenshot below*), create the table as a dynamic table 1069. Do this by selecting *Dynamic number of rows*. This will create a table with as many rows as there are corresponding row elements in the page source. Specify that the table has four columns and one header row (*see screenshot below*). Click **OK** to create the table.



New Table

Tables and row counts can be static or repeating.
For repeating tables or rows you have to assign a xml element to the table or row.

□ Table will be repeating (for every element occurrence 1 table will be created)

Columns

Static number of columns: 4

Rows

○ Static number of rows: 2

● Dynamic number of rows:

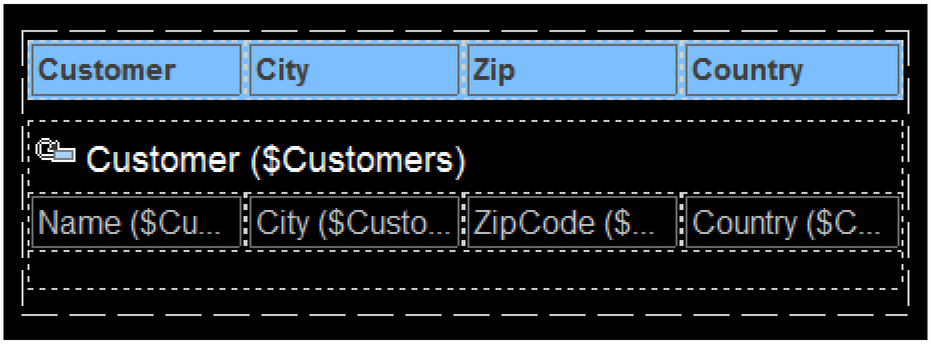
Header rows: 1

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

□ Automatic Append/Delete controls

3. From the Page Sources Pane 270, drag the `Customer` element to the **Repeating Row** icon of the table in the design. Each `customer` element will now correspond to one row of the table, and the `customer` element will be the XPath context node of the table.
4. Drag a Label control 554 from the Controls Pane 266 and drop it into the first column of the header row. Type in *Customer* as the text of the label (*see screenshot below*). Create headers for the other columns similarly: *City*, *Zip*, and *Country*.
5. Select all four labels (by pressing **Ctrl** while selecting each label), and apply whatever label formatting you like (via the Styles & Properties Pane 274).
6. Drag a Label control 554 from the Controls Pane 266 and drop it into the first column of the table-body row. Then, from the Page Sources Pane 270, drag the `Customer/Name` element of the `$CUSTOMERS` page source onto the label (*see screenshot below*). This label will display the contents of the customer's `Name` element.
7. Create the contents of the other table columns similarly: by dragging the `City`, `ZipCode`, and `Country` elements onto the respective labels (*see screenshot below*).



| Customer | City | Zip | Country |
|---|---|---|---|
| Customer ($Customers) | | | |
| Name ($Cu... | City ($Custo... | ZipCode ($... | Country ($C... |

8. Select all four labels (by pressing **Ctrl** while selecting each label), and apply whatever label formatting you like (via the Styles & Properties Pane 274).

## 4.6.5 Top Page: Action Group, Go to Sub Page

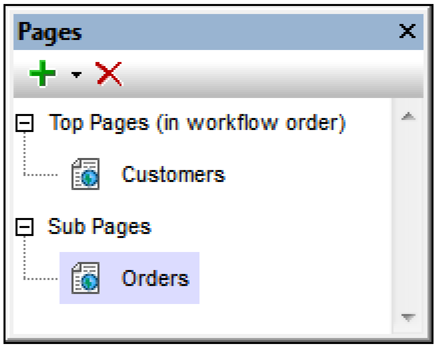
In the previous section[194], we created a table that displays the details of each customer in a separate table row. When the end user clicks on any customer detail (say name or city), we want to display that customer's current orders. We plan to do this by creating a sub page that will filter all current orders to display only those of the selected customer. When the end user selects a customer in the top page, the selection will be conveyed to the sub page via the shared `$XML1` page source. In the sub page, the selected-customer data is used in the **`Visible`** property of the table row group to filter the orders. We need therefore to do the following when the end user makes a selection by clicking a customer-detail label:

- Convey customer details of the clicked label to the `$XML1` tree (which is shared between top page and sub page).
- Go to the sub page, which will show the orders of the selected customer.

These are the actions that will need to be performed when any of the labels in a customer row is clicked. Since the same sequence of actions must be executed for each label, we can save the sequence of actions in a common action group[667], and then assign this action group to each label's `OnLabelClicked`[554] event. Before we define the sequence of actions we need to create the sub page that the Go to Subpage[783] action will target.

### Create the sub page

Click any item in the Pages Pane[257] and select **Add Sub Page**. Rename the added sub page to Orders (*see screenshot below*). You can double-click the name of the sub page to edit it.



### Create the action group

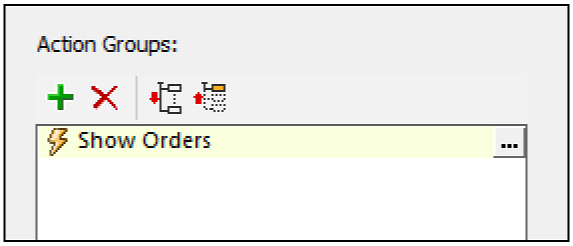
We will create an action group consisting of the following actions:

- Two Update Node[886] actions to update the two nodes of the `$XML1` page source.
- A Go to Subpage[783] action to go to the Orders sub page.

After we create the action group, we can assign it to the `OnLabelClicked`[554] event of each of the four table-body-row labels. At runtime, the action sequence will be executed when the end user clicks the label of any customer detail.

Create the action group as follows:

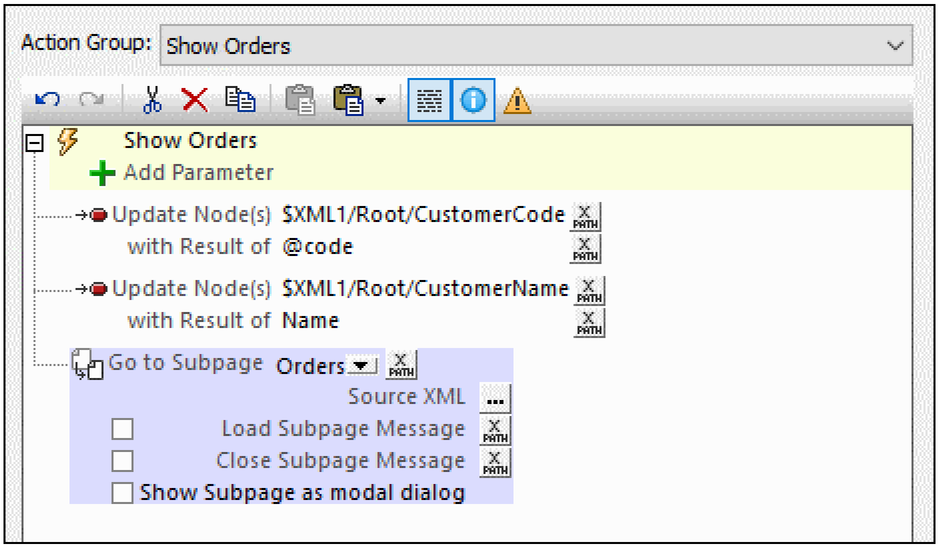
1. Right-click any of the four table-body-row labels and select **Control Actions for OnLabelClicked**.
2. In the Actions dialog[667] that appears, in the right-hand Action Groups pane, click the **Add a Group** button.
3. Rename the added group to *Show Orders* (*screenshot below*; you can double-click the name to edit it in the Action Groups dialog).



Action Groups:

Show Orders ...

4. Click the **Additional Dialog** button of *Show Orders* to display the Action Groups dialog (*screenshot below*).
5. Define the two Update Node[886] actions and the Go to Subpage[783] action as shown in the screenshot below, taking care to enter the XPath expressions exactly as shown. For the Go to Subpage[783] action, select the Orders sub page from the dropdown list of the combo box.



Action Group: Show Orders

Show Orders
Add Parameter
Update Node(s) $XML1/Root/CustomerCode
with Result of @code
Update Node(s) $XML1/Root/CustomerName
with Result of Name
Go to Subpage Orders
Source XML ...
□ Load Subpage Message
□ Close Subpage Message
□ Show Subpage as modal dialog

6. Click **OK** when done.

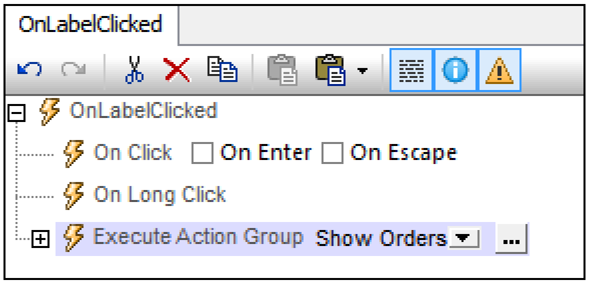
We have set the following updates: (i) the `$XML1//CustomerCode` element will be updated with the `Customer/@code` value of the selected customer (from `Customers.xml`[190]), and (ii) the `$XML1//CustomerName` element will be updated with the `Customer/Name` value of the selected customer (from `Customers.xml`[190]).

## Assign the action group to events

We now need to specify that the *Show Orders* action group is executed when a label is clicked. Do this as follows.

1. Right-click the Name label of the first column of the table-body row, and select **Control Actions for OnLabelClicked**.

2. In the Actions dialog[667] that appears, drag the *Show Orders* action group and drop it below the On Long Click event as shown in the screenshot below. Click **OK** to finish.



OnLabelClicked

- OnLabelClicked
  - On Click □ On Enter □ On Escape
  - On Long Click
  - Execute Action Group Show Orders ...

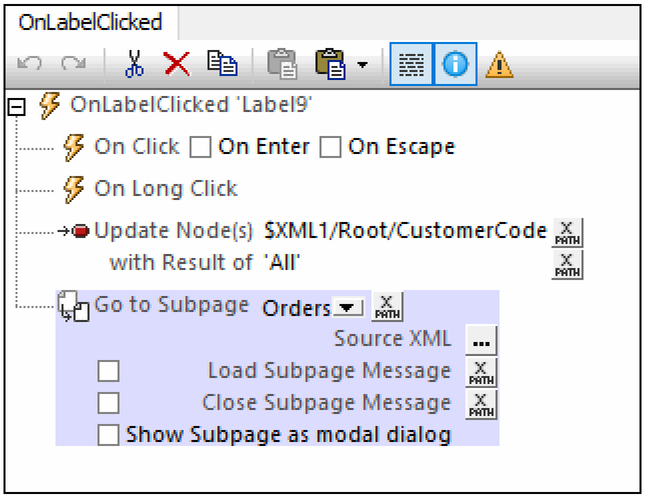
3. Repeat Step 2 for each of the three other labels of the table-body row. This ensures that the *Show Orders* sequence of actions will be executed when any of the four labels of a row is clicked.

## 4.6.6 Top Page: Show All Orders Action

In the previous section[196], we created a sequence of actions to perform when the end user clicks any one customer in the table of customers. In such an event, the orders of that customer will be displayed in the subpage. (We plan to do this by using the visibility property of the Orders table.) In this section, we will create a label that end users can click if they wish to see all the current orders (of all customers) in the `Orders.xml`[190] database.

Add the *Show all orders* label as follows:

1. Drag a Label control[554] from the Controls Pane[266] and drop it below the Customers table. Type in *Show all orders* as the text of the label (*see screenshot below*).
2. Format the label as you like with properties from the Styles & Properties Pane[274].
3. Open the Actions dialog[667] via the control's context menu command **Control Actions OnLabelClicked Action**, and add a sequence of actions for the `OnLabelClicked`[554] event (*see screenshot below*).



OnLabelClicked

- OnLabelClicked 'Label9'
  - On Click □ On Enter □ On Escape
  - On Long Click
  - Update Node(s) $XML1/Root/CustomerCode
    with Result of 'All'
  - Go to Subpage Orders
    Source XML ...
    □ Load Subpage Message
    □ Close Subpage Message
    □ Show Subpage as modal dialog

Note that the `$XML1/Root/CustomerCode` element has been defined to update to `'All'` if the end user clicks the *Show all orders* label.

4. Click **OK** to finish.

## 4.6.7 Sub Page: Page Sources

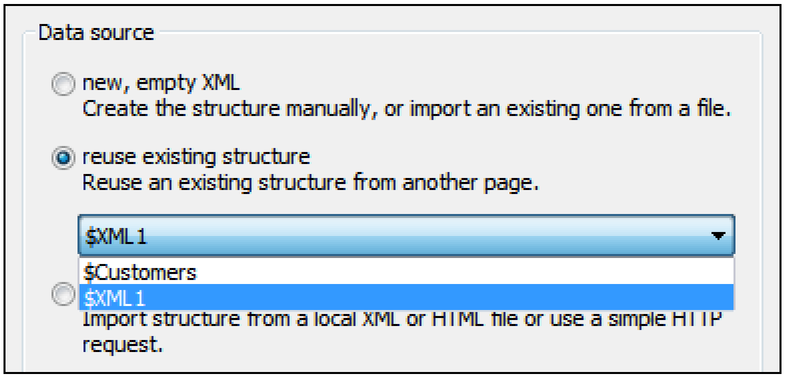
The sub page has already been created[196] because this step was needed earlier, in order to define the Go to Subpage[783] action. Our sub page is called Orders. It will have the following page sources:

- **`$ORDERS`**, which will have the structure and content of `Orders.xml`[190]. This page source is needed in order to display the orders contained in `Orders.xml`
- **`$XML1`**: This page source is shared with the top page. It is needed in the sub page because it contains information indicating which orders the end user wants to see. Specifically, it will contain the code of the customer that the user has selected on the top page.
- **`$CUSTOMERS`**: This page source is the same one that is used in the top page and it is shared with the top page. It is used in the sub page to retrieve customer information, such as the customer name.

### Adding the page sources

Add the three page sources as follows:

1. In the Page Sources Pane[270], click the **Add Source** icon, and select *Reuse existing structure*[317] *(see screenshot below)*.
2. In the option's combo box (*screenshot below*), select `$XML1`, and click **OK**.



Data source

○ new, empty XML
Create the structure manually, or import an existing one from a file.

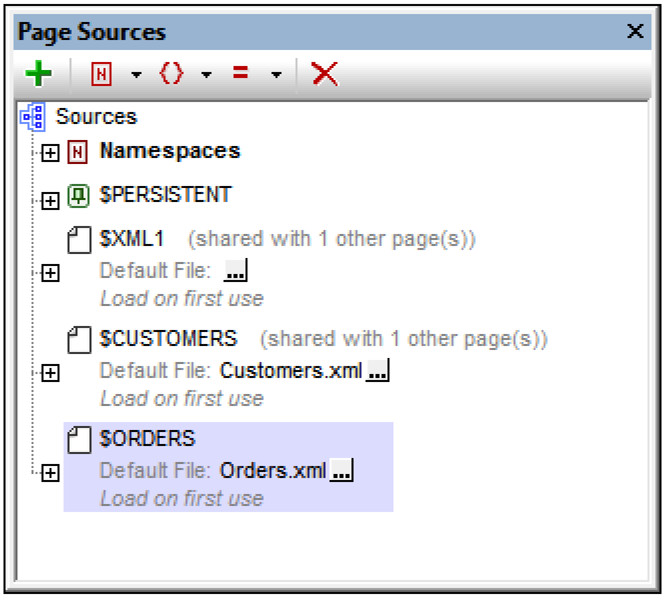
● reuse existing structure
Reuse an existing structure from another page.

$XML1 ▾
$Customers
$XML1

○ Import structure from a local XML or HTML file or use a simple HTTP request.

The `$XML1` source will be added. Next to its name will be an annotation saying that it is shared with one other page. Note that the structure and content of `$XML1`, as created in the top page, is already present.

3. Add the second shared page source, `$CUSTOMERS`, in the same way. Note that the structure, content, and default file will be as created in the top page.
4. In the Page Sources Pane[270], click the **Add Source** icon, and select *New XML or HTML structure imported from file*[317]. Browse for the file `Orders.xml`[190] and click **Open**. When you are prompted about whether to deploy the file, choose **Yes**. A page source called `$XML2` will be created.
5. Double-click the root node `$XML2` and edit the name to `$ORDERS` (*see screenshot below*).
6. Click the **Additional Dialog** button of the `$ORDERS` default file[355]. In the dialog that appears, select the *Relative paths* check box to make the file's path relative to the design (*see screenshot below*) and click **OK**.

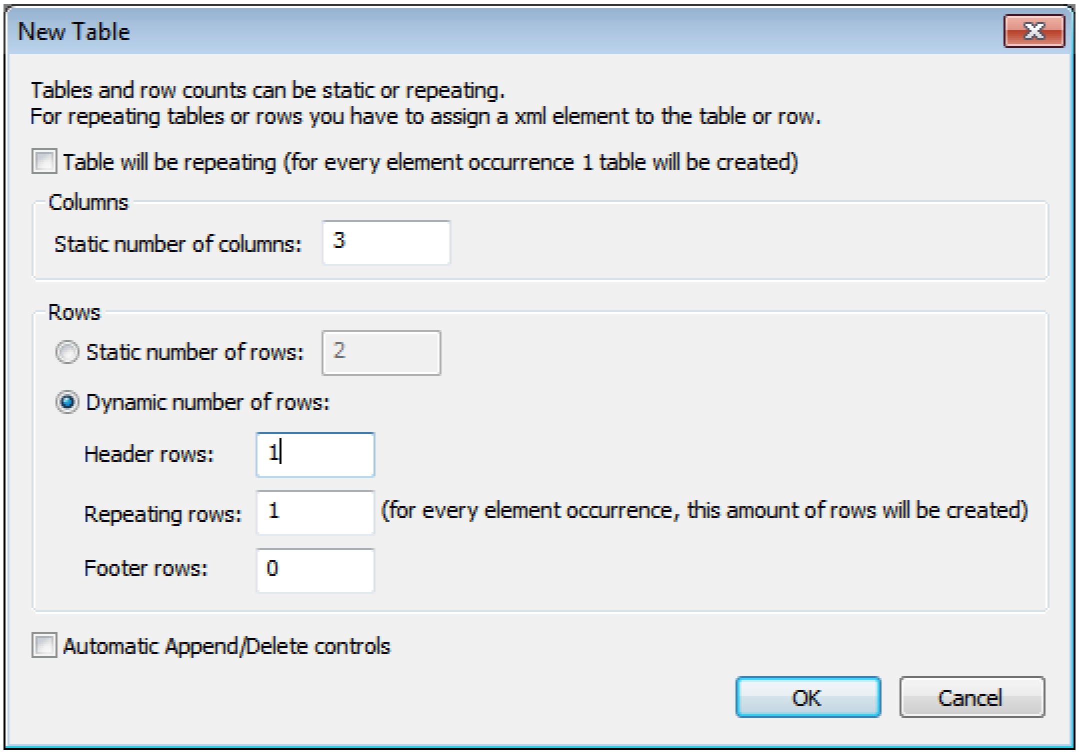

Page Sources
Sources
Namespaces
$PERSISTENT
$XML1 (shared with 1 other page(s))
Default File: ...
Load on first use
$CUSTOMERS (shared with 1 other page(s))
Default File: Customers.xml ...
Load on first use
$ORDERS
Default File: Orders.xml ...
Load on first use

After the page sources have been added, you are ready to create the design of the sub page.

## 4.6.8 Sub Page: Orders Table

We will now create a table to display the orders of the customer that is selected on the top page by the end user. The orders are stored in the XML page source `Orders.xml` 190. Create the Orders table as follows.

1. Drag a Table control 614 from the Controls Pane 266 and drop it into the design.
2. In the New Table dialog that appears (*see screenshot below*), create the table as a dynamic table 1069. Do this by selecting *Dynamic number of rows*. This will create a table with as many rows as there are corresponding row elements in the page source. Specify that the table has three columns and one header row (*see screenshot below*). Click **OK** to create the table.



New Table

Tables and row counts can be static or repeating.
For repeating tables or rows you have to assign a xml element to the table or row.

☐ Table will be repeating (for every element occurrence 1 table will be created)

Columns

Static number of columns: 3

Rows

○ Static number of rows: 2

● Dynamic number of rows:

Header rows: 1

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

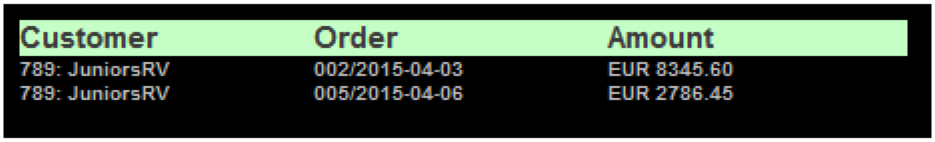
☐ Automatic Append/Delete controls

3. From the Page Sources Pane[270], drag the `Order` element to the **Repeating Row** icon of the table in the design. Each `order` element will now correspond to one row of the table, and the `order` element will be the XPath context node of the table.
4. Drag a Label control[554] from the Controls Pane[266] and drop it into the first column of the header row. Type in *Customer* as the text of the label (*see screenshot below*). Create headers for the other columns similarly: *Order* and *Amount*.
5. Select all three header labels (by pressing **Ctrl** while selecting each label), and apply whatever label formatting you like in the Styles & Properties Pane[274].
6. Drag Label controls[554] from the Controls Pane[266] and drop them, respectively, into the three columns of the table-body row.
7. Select all three table-body row labels (by pressing **Ctrl** while selecting each label), and apply whatever label formatting you like in the Styles & Properties Pane[274].

The labels of the table-body rows have now been placed in the table cells. Their text will be specified via the XPath expressions described in the next section.

## Create XPath expressions for the label texts

The table output we want is shown below. Notice the contents of the different columns.



| Customer | Order | Amount |
|---|---|---|
| 789: JuniorsRV | 002/2015-04-03 | EUR 8345.60 |
| 789: JuniorsRV | 005/2015-04-06 | EUR 2786.45 |

To create XPath expressions for a label's text, first select the label. In the Styles & Properties Pane[274], select the label's `Text` property and click its XPath icon. (Alternatively, you can: (i) right-click the `Text` property and select **Calculate with XPath**, or (ii) select the `Text` property and click the XPath icon in the pane's menu bar.) In the XPath dialog[1244] that appears, enter the respective XPath expressions. Note that the XPath context node is the respective `$ORDERS/Orders/Order` element.

*For the Customer column*

```
if ($XML1/Root/CustomerCode!='All')
then concat(CustomerCode, ': ', $XML1/Root/CustomerName)
else concat(CustomerCode, ': ', for $i in CustomerCode return
$CUSTOMERS/Customers/Customer[@code=$i][1]/Name)
```

- For a table with orders of a selected customer, the customer name is obtained from the `$XML1` tree.
- For a table showing all orders, the customer name is retrieved from the `$CUSTOMERS` tree by using the customer code in the `$ORDERS` tree as the key. (The customer code is present in both trees.)

*For the Order column*

```
concat(@number, '/', OrderDate)
```

*For the Amount column*

```
concat(Currency, ' ', OrderAmount)
```

### Finishing the Orders table

After having defined the contents of each column, format the labels as you like with properties from the Styles & Properties Pane[274]. Now we have to specify the visibility property of the table row group so that only the customer that has been selected on the top page will be displayed in the table. The visibility property is described in the next section[202].

## 4.6.9 Sub Page: Visibility Property

The Orders table that we have created[200] in the Orders sub page is a dynamic table that generates one row for each `Order` element (or record) in the page source `Orders.xml`[190]. The `Order` elements are presented in the order in which they occur in the data file. But we can control which `Order` elements are displayed. This is done with the `Visible` property of the Table Row Group. The property takes an XPath expression that selects the `Order` elements to display

To set the XPath expression of the `Visible` property, select the repeating row in the design, and, in the Styles & Properties Pane[274], go to the properties of the Table Row Group, and click the **XPath** icon of the `Visible` property. In the Edit XPath/XQuery Expression dialog[1244] that appears, enter the following XPath expression:

```
if ($XML1/Root/CustomerCode!='All') then CustomerCode=$XML1/Root/CustomerCode else
CustomerCode
```

This XPath expression works as follows:

1. The `if` clause of the expression tests whether the element `$XML1/Root/CustomerCode` contains the string `All`.

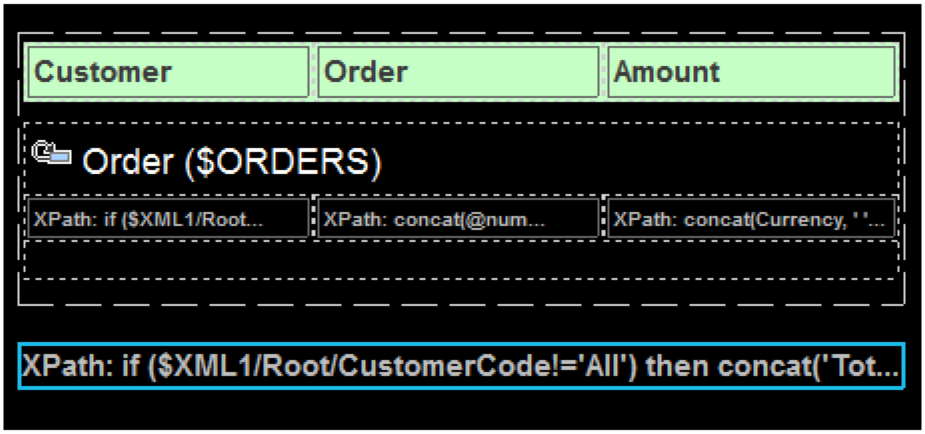
2. If the element `$XML1/Root/CustomerCode` **does not contain** the string `All`, then all `Order` elements that have their `CustomerCode` element content equal to the content of the `$XML1/Root/CustomerCode` element will be selected. In effect, these will be the `Order` elements of the customer that was selected by the end user. Remember that the customer's `CustomerCode` has been stored in the `$XML1` page source (*see Top Page: Action Group, Go to Sub Page* 196).
3. If the element `$XML1/Root/CustomerCode` **contains** the string `All`, then all `Order` elements that have a child `CustomerCode` element will be selected. In effect, this will select all `Order` elements in the data file.

**Note:** The advantage of using the `Visible` property is that it is a simple, efficient, and effective alternative to other ways of generating a table containing selected elements only.

## 4.6.10 Sub Page: Decimal Totals in XPath

To complete the design, we will add a label that displays the total amount of the displayed orders. Do this as follows:

1. Drag a Label control 554 from the Controls Pane 266 and drop it below the Orders table (*see screenshot below*).
2. In the Styles & Properties Pane 274, Click the **XPath** icon of the control's `Text` property.
3. In the In the Edit XPath/XQuery Expression dialog 1244 that appears, enter the XPath expression to calculate the total amounts (*the expression is given below*), and click **OK**.


Customer
Order
Amount
Order ($ORDERS)
XPath: if ($XML1/Root...
XPath: concat(@num...
XPath: concat(Currency, ' '...
XPath: if ($XML1/Root/CustomerCode!='All') then concat('Tot...

### The XPath expression to calculate the total amount

We need to calculate totals in two events: (i) for the orders of the selected customer, and (ii) for all orders. This can be done with the following XPath expression:

```
if ($XML1/Root/CustomerCode!='All')
then concat('Total: ', xs:decimal(sum
($ORDERS//Order[CustomerCode=$XML1/Root/CustomerCode]/OrderAmount)))
else concat('Total: ', xs:decimal(sum ($ORDERS//OrderAmount)))
```

This XPath expression works as follows:

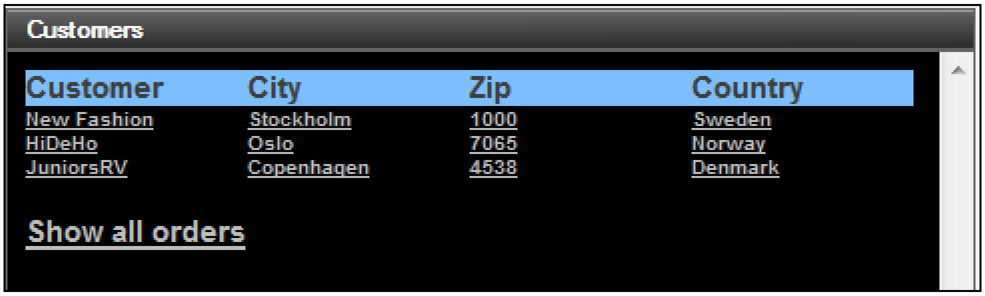
1. The `if` clause of the expression tests whether the element `$XML1/Root/CustomerCode` contains the string `All`.
2. If the element `$XML1/Root/CustomerCode` **does not contain** the string `All`, then the `OrderAmount` element of all `Order` elements that have their `CustomerCode` element content equal to the content of the `$XML1/Root/CustomerCode` element will be selected. These will be the amounts of those `Order` elements of the customer that was selected by the end user. Remember that the customer's `CustomerCode` has been stored in the `$XML1` page source (*see Top Page: Action Group, Go to Sub Page* 196).
3. If the element `$XML1/Root/CustomerCode` **contains** the string `All`, then all `OrderAmount` elements will be selected.

The selected OrderAmount elements are summed using the `sum()` function of XPath. Since the `sum()` function uses the `xs:double` type and returns an `xs:double` number, the amount will have more than the two decimal places we require in a currency. We therefore use the `xs:decimal` type converter to round the `xs:double` to a two-decimal place number.

## 4.6.11 Simulation and Testing

After completing the design, run a simulation (by pressing **F5**) and test your design. The screenshots below show the top page and sub pages in the MobileTogether Designer simulator.

*Top page: Customers*

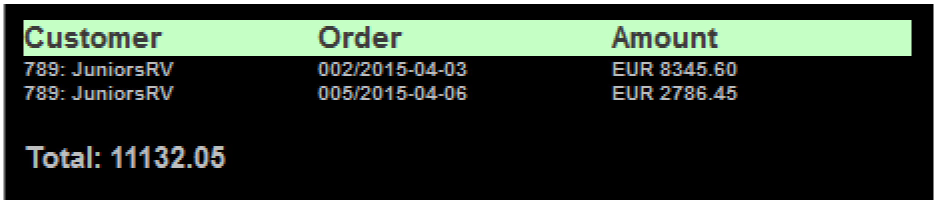


Customers

| Customer | City | Zip | Country |
|---|---|---|---|
| New Fashion | Stockholm | 1000 | Sweden |
| HiDeHo | Oslo | 7065 | Norway |
| JuniorsRV | Copenhagen | 4538 | Denmark |

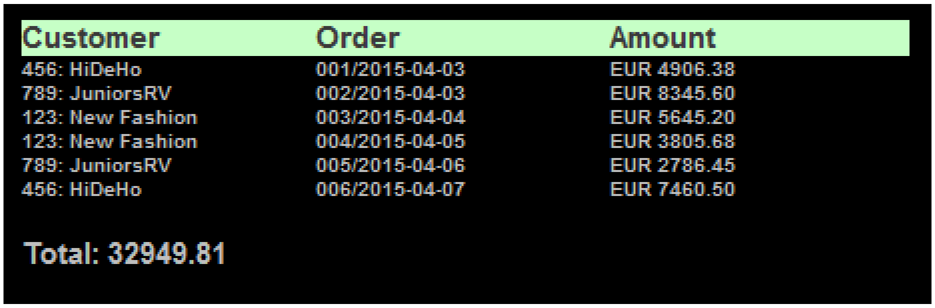
Show all orders

*Sub page: Orders (for selected customer and all customers, respectively)*



| Customer | Order | Amount |
|---|---|---|
| 789: JuniorsRV | 002/2015-04-03 | EUR 8345.60 |
| 789: JuniorsRV | 005/2015-04-06 | EUR 2786.45 |

Total: 11132.05



| Customer | Order | Amount |
|---|---|---|
| 456: HiDeHo | 001/2015-04-03 | EUR 4906.38 |
| 789: JuniorsRV | 002/2015-04-03 | EUR 8345.60 |
| 123: New Fashion | 003/2015-04-04 | EUR 5645.20 |
| 123: New Fashion | 004/2015-04-05 | EUR 3805.68 |
| 789: JuniorsRV | 005/2015-04-06 | EUR 2786.45 |
| 456: HiDeHo | 006/2015-04-07 | EUR 7460.50 |

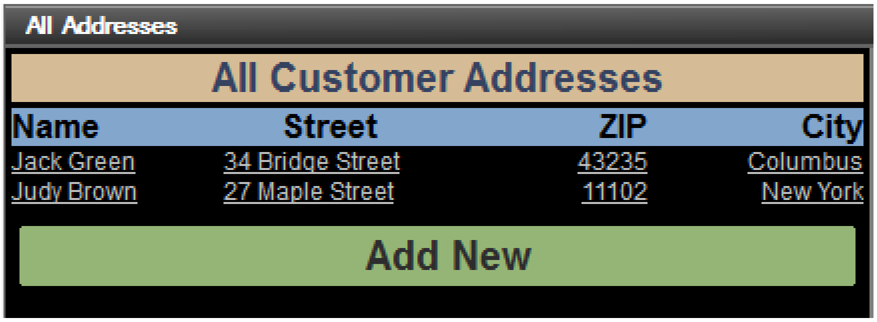
Total: 32949.81

That's it!

# 4.7 Add and Edit Records

The objectives of this tutorial are as follows:

- Add a new customer record to a customer database
- Edit a record in the database



All Addresses

All Customer Addresses

| Name | Street | ZIP | City |
|---|---|---|---|
| Jack Green | 34 Bridge Street | 43235 | Columbus |
| Judy Brown | 27 Maple Street | 11102 | New York |

Add New

## Design plan

The design is constructed in this way:

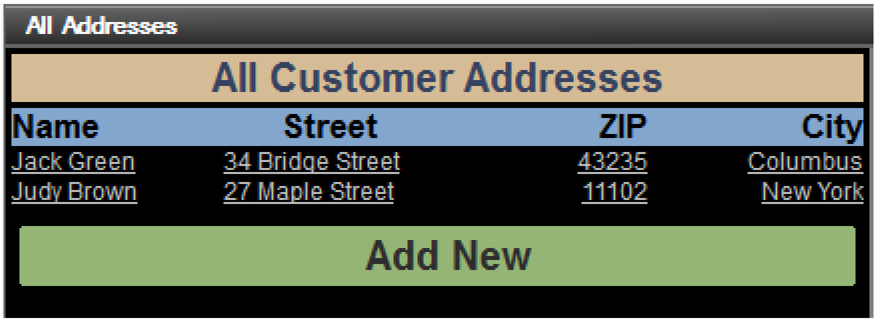
- Two XML page sources are used. The `$PERSISTENT` tree is used to hold the customer database. A `$EDIT` tree is used to hold the one record (a `Customer` element) that is being currently edited.
- The `$EDIT/Customer` element is created when an **Add New** button is clicked.
- When a **Save** button is clicked, the `$EDIT/Customer` element is appended as the last element of the `$PERSISTENT` tree. The record is then deleted from the `$EDIT` tree (so that a record can be loaded subsequently, either new or for editing).
- The customer database is displayed as a table that shows the `Customer` elements in the `$PERSISTENT` tree (*see screenshot above*).
- When any field of a record in the customer database is clicked, then that record is loaded in the `$EDIT` tree, where it can be edited.
- When the Save button is clicked after a record has been edited in this way, it is saved back to the `$PERSISTENT` tree at its original location.

## The tutorial file

The finished tutorial file is located in your ([72]My) Documents[72] MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\**`AddEditRecords\AddAndEditRecords.mtd`. Open this file and check the design settings in it while reading through this tutorial.

## 4.7.1 Design Pages

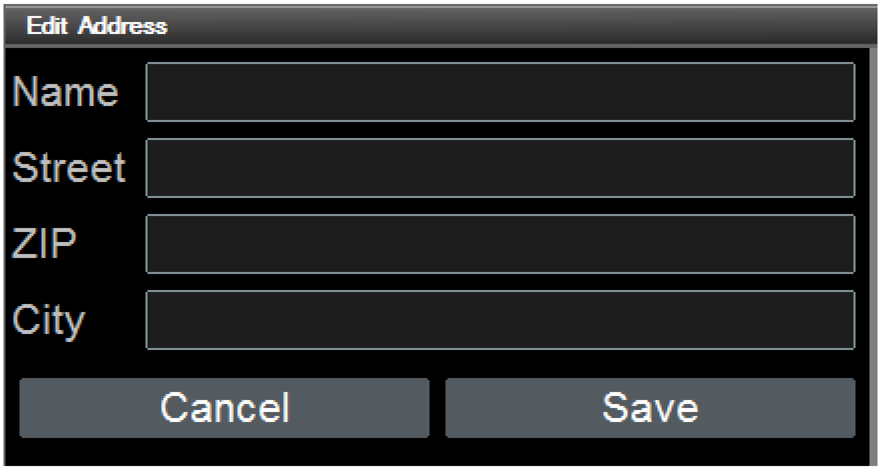
When the solution starts, the main (top) page, named *All Addresses*, is displayed. This page contains: (i) a table displaying all the records in the customer database (contained in the `$PERSISTENT` tree), and (ii) an **Add New** button that enables the user to add new records to the customer database.



All Customer Addresses

| Name | Street | ZIP | City |
|---|---|---|---|
| Jack Green | 34 Bridge Street | 43235 | Columbus |
| Judy Brown | 27 Maple Street | 11102 | New York |

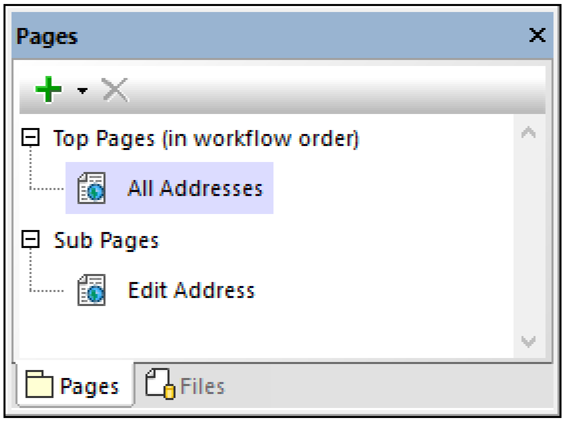
When the user clicks the **Add New** button, the button's action takes it to a sub page called *Edit Address (screenshot below)*. This page displays a data entry form in which the user can enter details of a new customer. The sub page is required so that the workflow can move between two cleanly separated page designs.



After the new-customer data is entered in the *Edit Address* page, the user can click **Save**. Alternatively the user can click **Cancel** to abort. When either button is clicked, the workflow exits the sub page and goes back to the main page. For an explanation of the actions that are carried out when these buttons are clicked, see the section Add a New Record[209].

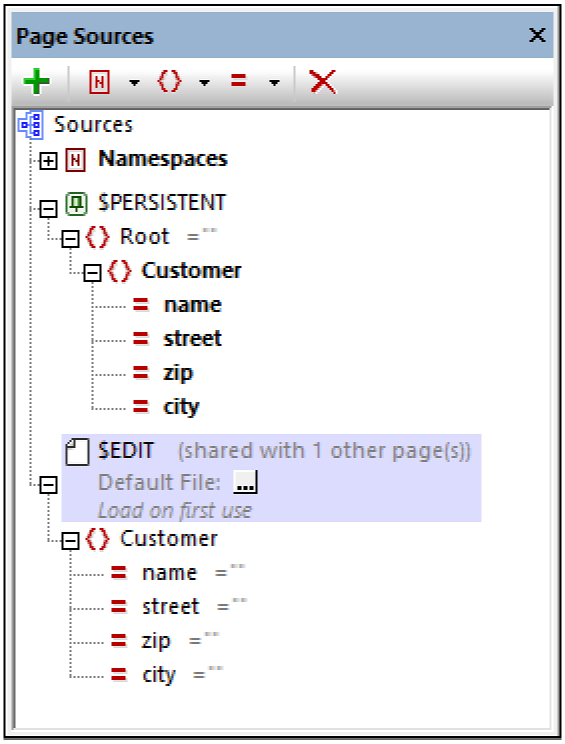
## Creating the pages

The top page and sub page are created in the Pages pane[257] (*screenshot below*).



## 4.7.2 Page Sources

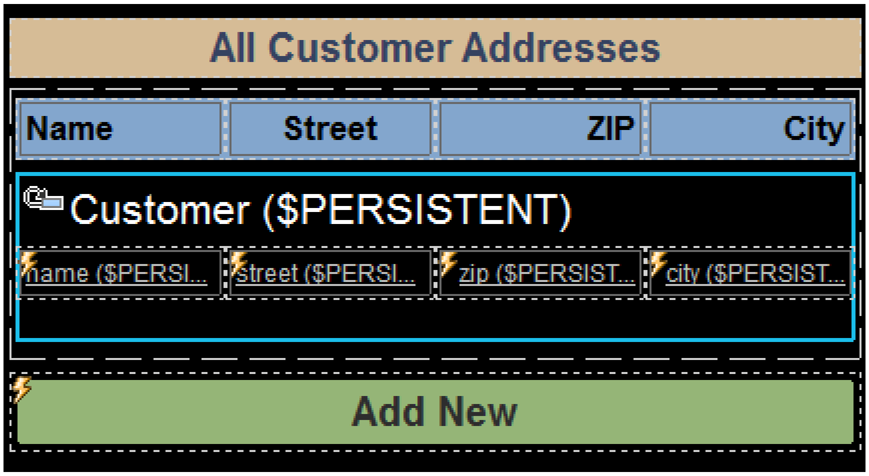
The records of the customer database is stored in the `$PERSISTENT` tree. When a new record is added or an existing record is edited, then this record is loaded in the `$EDIT` tree. In both trees, each record is stored in a `Customer` element. The two trees (*see screenshot below*) are built manually by using the toolbar icons of the Page Sources pane[270] or the context menus of nodes in the tree.



Note the following points:

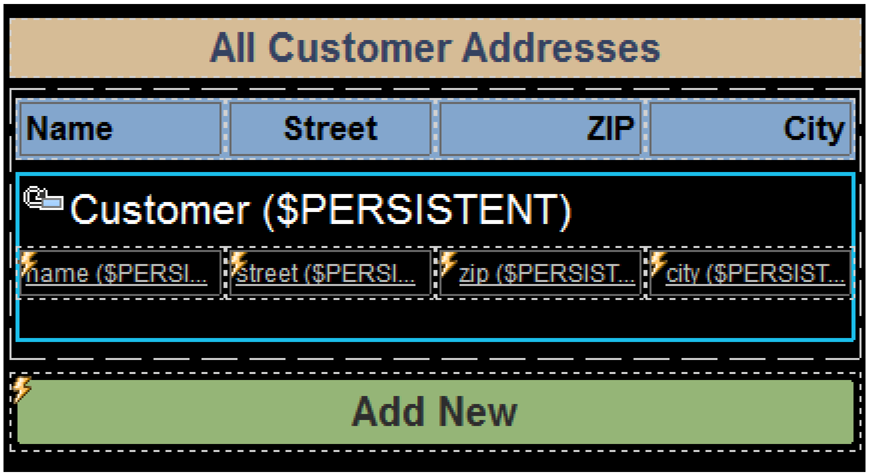
- In both trees, a `Customer` element corresponds to a single customer record.

- The data of each customer is stored in the attributes of that customer's `Customer` element.
- The `$EDIT` tree is shared between both pages of the design. This means that the data available in the tree is commonly available on both pages.
- In the `$EDIT` tree, each of these attributes is set to have a fixed value of an empty string when the tree is loaded. This setting, **Ensure exists on load**, is available in the context menu of each attribute (obtained by right-clicking the attribute). The reason for toggling this setting on is this: Each time the **Add New** button is clicked, we want to (re)load this tree with empty attribute values. The new customer's details can thus be added to an empty customer record.
- The reason why the `Customer` element and its child attributes are shown in bold is because they have been created as page source links[403] in the design: they are used to display the customer database in the columns of the table on the main page (*see screenshot below*).

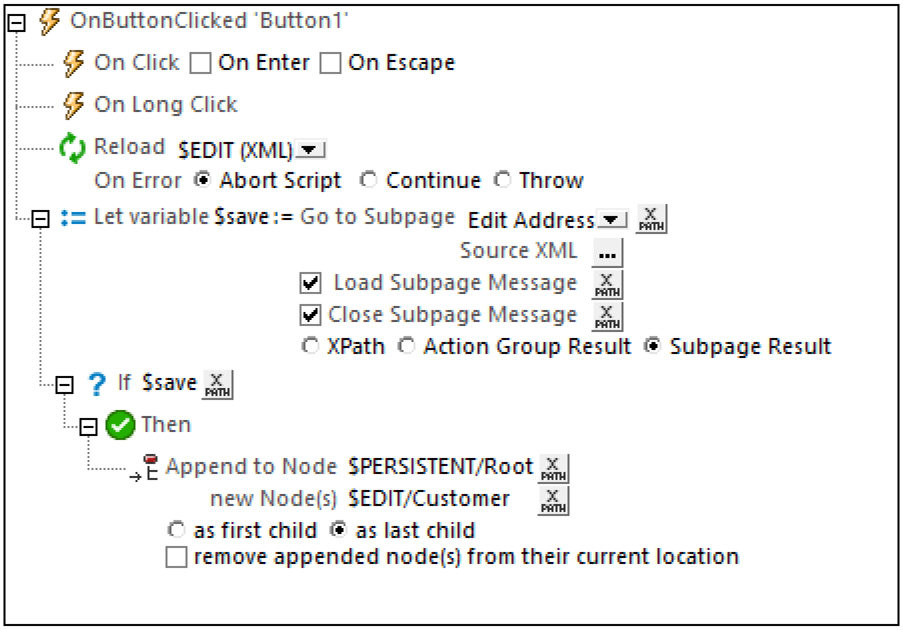


## 4.7.3 Add a New Record

When the user opens the solution, the customer database is empty. To add a record, the **Add New** button (*see screenshot below*) is clicked.



The **Add New** button has two `onButtonClicked` actions (Reload[801] and Let[900]), which are shown in the screenshot below.



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

Reload $EDIT (XML)

On Error ● Abort Script ○ Continue ○ Throw

Let variable $save := Go to Subpage Edit Address

Source XML ...

☑ Load Subpage Message

☑ Close Subpage Message

○ XPath ○ Action Group Result ● Subpage Result

If $save

Then

Append to Node $PERSISTENT/Root

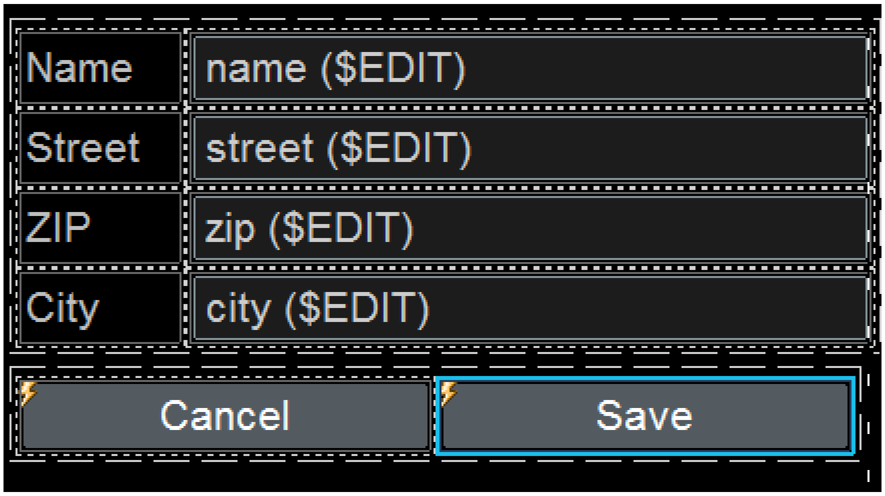
new Node(s) $EDIT/Customer

○ as first child ● as last child

□ remove appended node(s) from their current location

These two actions carry out the following sequence of steps:

1. The Reload action[801] reloads the `$EDIT` tree. Since the nodes of this tree have been defined to be loaded with a fixed value of the empty string[208], all the fields of the customer record will be empty.
2. The Let action[900] creates a variable called `$save`, which goes to the sub page *Edit Address* (*screenshot below*) and fetches its result.



Name name ($EDIT)

Street street ($EDIT)

ZIP zip ($EDIT)

City city ($EDIT)

The result of the sub page is returned when the **Save** button of the page is clicked. The **Save** button executes the Close Subpage action[788] and returns its result—which is the `Customer` node. So this node is then stored in the `$save` variable.

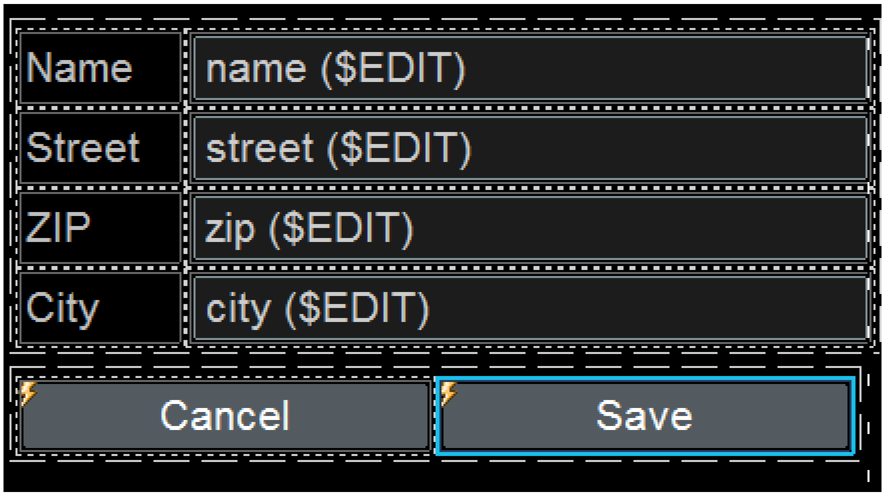
3. An If-Then action[894] next checks whether the `$save` variable exists.
4. If the `$save` variable exists, then the **Then** clause of the action is executed. This causes the `$EDIT/Customer` element to be appended (by using the Append Node(s)[875] action) as the last child

node of the `$PERSISTENT/Root` element. In this way, when new-customer data that is added in the *Edit Address* sub page is saved, the entire customer record is appended as the last record of the customer database in the `$PERSISTENT` tree.

5. The **Cancel** button executes the Close Subpage action 788 without returning a result. The effect is to return to the main page without modifying the customer database in any way.

## 4.7.4 Enter New-Record Data

The *Edit Address* (sub) page (*screenshot below*) is used to enter new-customer data as described below:

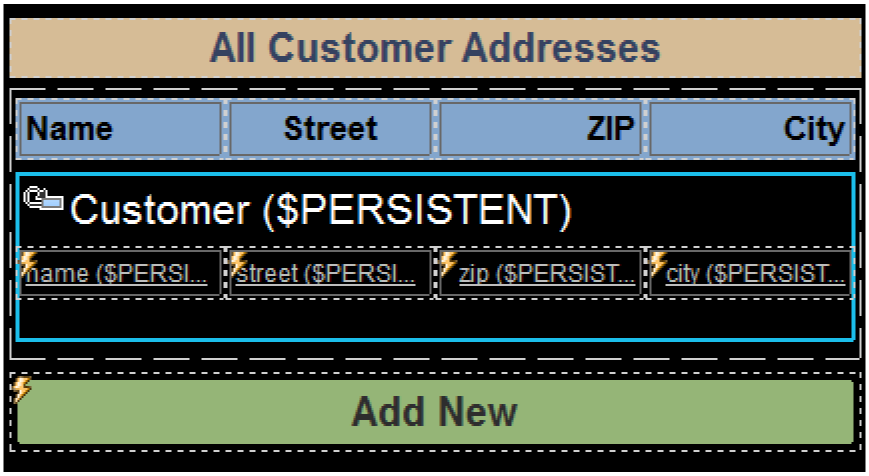


| | |
|---|---|
| Name | name ($EDIT) |
| Street | street ($EDIT) |
| ZIP | zip ($EDIT) |
| City | city ($EDIT) |

- Each customer data edit field 495 is associated with a page source link 403 that is a node in the `$EDIT` tree.
- When the **Save** button is clicked, the entire `$EDIT/Customer` element is appended as the last child of the `$PERSISTENT/Root` element. The mechanism used to do this is explained in the previous section, Add a New Record 209.
- The **Cancel** button closes the sub page without returning any result. This is defined via the Close Subpage action 788 of the **Cancel** button.

## 4.7.5 Display All Records

The records of the customer database which is stored in the `$PERSISTENT` tree are displayed in a table with dynamic rows 1069 (*see screenshot below*).



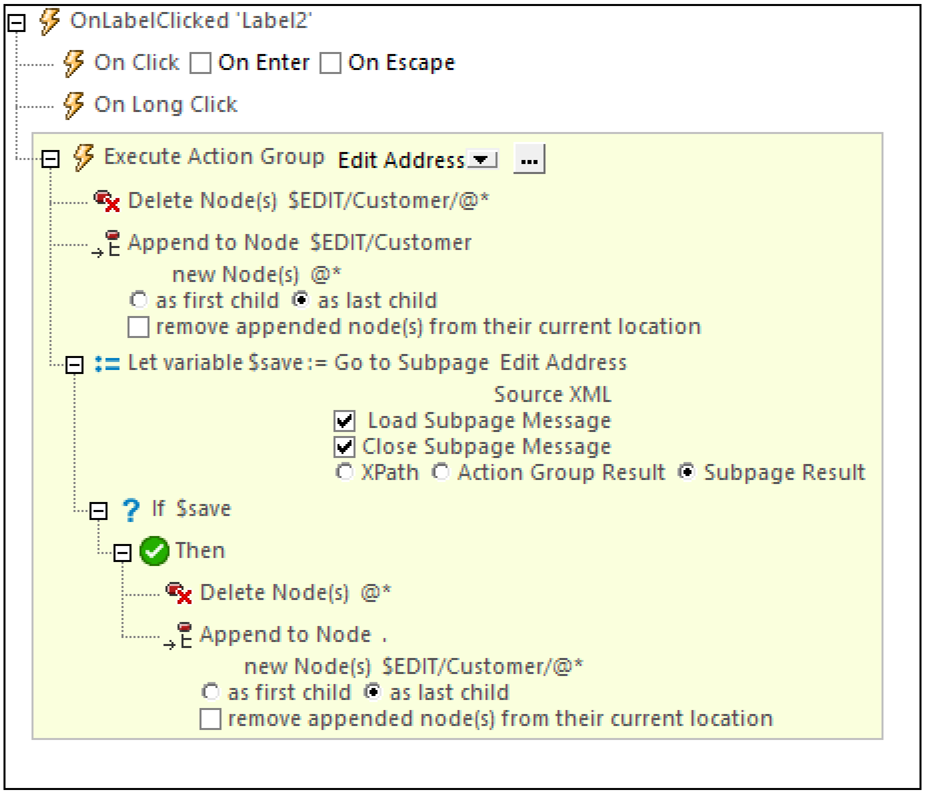
The table is defined as follows:

- The table has a single header row that is located outside the dynamic row.
- The dynamic row—that is, the row that repeats—is linked to the `$PERSISTENT/Root/Customer` page source element. As a result, in the solution, a new row is created for each **`Customer`** element.
- The cell of each table column contains a label control 554 that has been linked, respectively, to the different attribute nodes of the **`Customer`** element: **`name`**, **`street`**, **`zip`**, and **`city`**.
- Each label has the same set of actions defined for its `OnLabelClicked` event (*screenshot below*). These actions enable every customer record to be edited individually and to be saved back to the customer database (*see next section, Edit an Existing Record* 214). Since the set of actions is the same for all four labels, the actions have been defined in a single Action Group 940 that is reused on all four each labels.

## Action Group to edit addresses

The Action Group 940 that is added for the `OnLabelClicked` event of each label is shown in the screenshot and described below.

**Note:** The node within which the Action Group 940 has been added is the `$PERSISTENT/Root/Customer` node. So this is the context node of all XPath expressions in the Action Group 940.



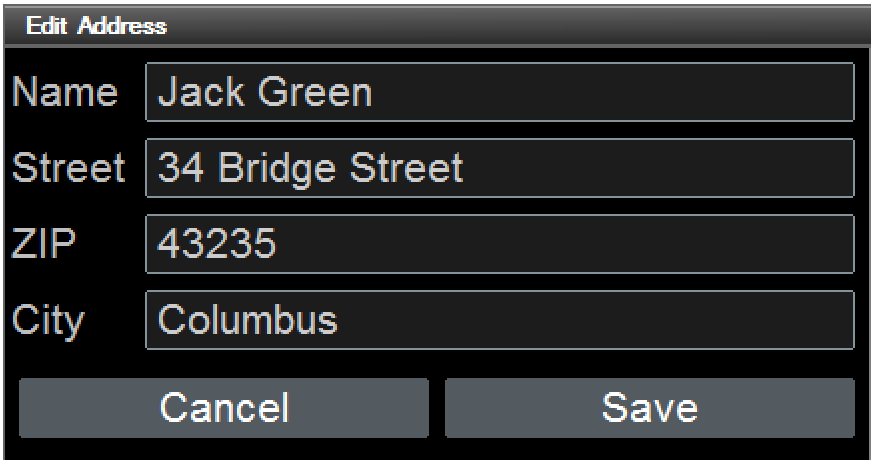
OnLabelClicked 'Label2'
On Click □ On Enter □ On Escape
On Long Click
Execute Action Group Edit Address
Delete Node(s) $EDIT/Customer/@*
Append to Node $EDIT/Customer
new Node(s) @*
○ as first child ● as last child
□ remove appended node(s) from their current location
Let variable $save := Go to Subpage Edit Address
Source XML
☑ Load Subpage Message
☑ Close Subpage Message
○ XPath ○ Action Group Result ● Subpage Result
If $save
Then
Delete Node(s) @*
Append to Node .
new Node(s) $EDIT/Customer/@*
○ as first child ● as last child
□ remove appended node(s) from their current location

The actions in this action group do the following:

- The Delete Node(s) [879] action deletes all the child attribute nodes of `$EDIT/Customer`. Remember that this page source contains the one customer record that is currently being edited [208].
- The deleted attribute nodes of the `$EDIT/Customer` node are replaced by the attribute nodes of the record that we want to edit. These nodes are the attribute nodes of the current context node: `$PERSISTENT/Root/Customer`. This replacement is done via the Append Node(s) [875] action.
- The Let action [900] creates a variable called `$save`, which goes to the sub page *Edit Address* (*screenshot below*) and fetches its result. The result of the sub page is returned when the **Save** button of the page is clicked. The **Save** button executes the Close Subpage action [788] and returns its result—which is the `Customer` node. So this node is then stored in the `$save` variable.
- An If-Then action [894] next checks whether the `$save` variable exists.
- If the `$save` variable exists, then the **`Then`** clause of the action is executed. This causes the attribute nodes of the current **`Customer`** element of the customer database to be deleted and for the attribute nodes from the `$EDIT/Customer` element to be appended to the current customer record of the customer database (by using the Append Node(s) [875] action). In this way, the edited customer data replaces the old customer data in the customer database.
- If the **Cancel** button is clicked, then the Close Subpage action [788] is executed without returning a result. The effect is to return to the main page without modifying the customer database in any way.

## 4.7.6 Edit an Existing Record

When the end user clicks a field in a customer record, the solution goes to the *Edit Address* sub page (*screenshot below*), where the user can edit that specific record.



Edit Address

Name: Jack Green

Street: 34 Bridge Street

ZIP: 43235

City: Columbus

- On clicking **Save**, the edited record is saved back to the customer database, and the solution goes back to the main page.
- If the user clicks Cancel, the original record is left unchanged, and the solution goes back to the main page..

The `OnButtonClicked` actions of these two buttons are as described in the section Add a New Record[209]. Note that this sub page is called when (i) a new record is to be added, or (ii) when a record is to be modified. Data modifications in both cases are saved to the `$EDIT` tree. The button actions are the same for both cases.

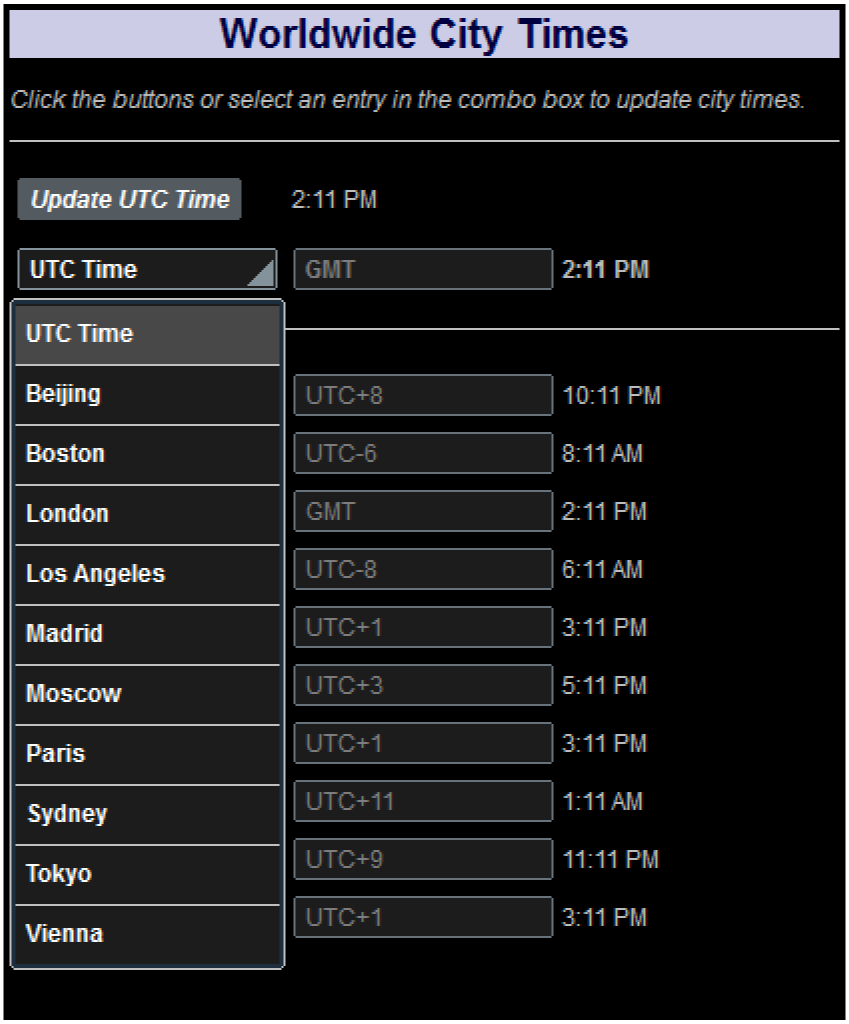
For the details of how the modified data replaces the old customer data in the customer database, see the description of the Action Group that carries out the relevant actions[212].

# 4.8 SOAP Requests

This tutorial describes how a design (`CityTimesViaSOAP.mtd`) has been constructed that uses SOAP-delivered data. The design generates SOAP requests from a WSDL file (`TimeService.wsdl`). The requests are sent to a web service (`http://www.nanonull.com/TimeService`), and the service's SOAP responses are used to update nodes in the design's XML tree.

The web service provides (i) the current UTC time, and (ii) the current time in a specific timezone; the timezone is submitted as a parameter of the relevant SOAP request. The aim of our design is to provide an interface for updating (i) the UTC time, and (ii) the time in selected cities. For the UTC time, a straight SOAP request (with no parameter) is sent to the web service, and the response is used to update an XML node. For city times, the city's timezone is submitted as a parameter of the SOAP request. Because nodes are updated with the responses and because the updated nodes are the page source links 404 of certain controls, the updated times are displayed immediately in the solution.

The interface looks something like the screenshot below. The lower part of the screen contains a list of selected cities. Clicking the *Update UTC Time* button and the <*City*> buttons updates the respective times (*see screenshot*). Selecting a city in the combo box also updates that city's time in the display. City time are also updated automatically when the page refreshes. This mechanism is described in the section Refreshing the Page 226.



Worldwide City Times

*Click the buttons or select an entry in the combo box to update city times.*

2:11 PM

GMT **2:11 PM**

| | |
|---|---|
| UTC+8 | 10:11 PM |
| UTC-6 | 8:11 AM |
| GMT | 2:11 PM |
| UTC-8 | 6:11 AM |
| UTC+1 | 3:11 PM |
| UTC+3 | 5:11 PM |
| UTC+1 | 3:11 PM |
| UTC+11 | 1:11 AM |
| UTC+9 | 11:11 PM |
| UTC+1 | 3:11 PM |

## The tutorial files

The files for this tutorial are located in your (72 My) Documents 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\SOAPRequests`**.

- `CityTimes.xml`: This is an XML data file that contains a list of cities and their timezones. It is used for structuring the data required by the design.
- `TimeService.wsdl`: This is the WSDL file from which the SOAP requests for the web service are generated.
- `CityTimesViaSOAP.mtd`: This is the completed MobileTogether design file. Open this file and refer to it while reading this tutorial. You can run a simulation in MobileTogether Designer by pressing **F5**.

The paths in the design file are relative, and the XML and WSDL data files have not been deployed to a server. So, if you copy these three files to any folder, you will be able to correctly run the simulations in MobileTogether Designer.

## Tutorial structure

This tutorial is organized into the following sections:

- The XML Page Source[217] describes the XML page source that is used for the structure and data of the design.
- Design Components[221] describes the various controls and actions of the design.
- Refreshing the Page[226] shows how values in the display can be automatically updated by actions defined for a page refresh.

## 4.8.1 The XML Page Source

We need one XML page source (`$XML1`) to hold and structure the data that is needed for the design. We will use the XML file `CityTimes.xml`, which is structured into the following parts (*see listing below*[218]):

- The **`UTC`** element, which is updated with the UTC time via a SOAP request when the user presses a button (*see listing below*[218]). This node is used to display the UTC time to the user.
- The **`RefreshTime`** element (*see listing below*[218]), which is designed to hold the time, in seconds, between automatic page refreshes. The user can select the value of this node.
- The **`SelectCity`** element, which contains the details (**`Name`**, **`TimeZone`**, and **`Time`**) of the city that the user selects (*see listing below*[218]). When the user selects a city from the dropdown list of a combo box, the **`SelectCity`** element's **`Name`** child element is updated with the name of the selected city. The **`SelectCity/TimeZone`** element is updated from the **`Cities`** database when the combo box selection is made, and **`SelectCity/TimeZone`** element is updated by the web service's response to the SOAP request that is sent when the user selects a city.
- The **`Cities`** element is a database containing the details (**`Name`**, **`TimeZone`**, and **`Time`**) of selected cities (*see listing below*[218]). (You can add more cities if you like.) The web service that we will be accessing requires a city's timezone in order to calculate and return the current time in the selected city. The database therefore must contain the timezone information. The **`Time`** child element of these **`City`** elements are used to hold a city's current time, which is obtained from the web service in response to a SOAP request. As with the UTC time above, the SOAP request is sent when the user presses a button (or selects a city in the combo box).

*Short version of the XML page source CityTimes.xml, showing the document's structure*

```
<CityTime xmlns="http://www.Nanonull.com/TimeService/">
        <UTC>12:00 AM</UTC>
        <RefreshTime>60</RefreshTime>
        <SelectCity>
                <City>
                        <Name>UTC Time</Name>
                        <TimeZone>GMT</TimeZone>
                        <Time>12:00 AM</Time>
                </City>
        </SelectCity>
        <Cities>
                <City>
                        <Name>Beijing</Name>
```

```
            <TimeZone>UTC+8</TimeZone>
            <Time>12:00 AM</Time>
        </City>
        ...
    </Cities>
```

### *Full listing of the XML page source, CityTimes.xml*

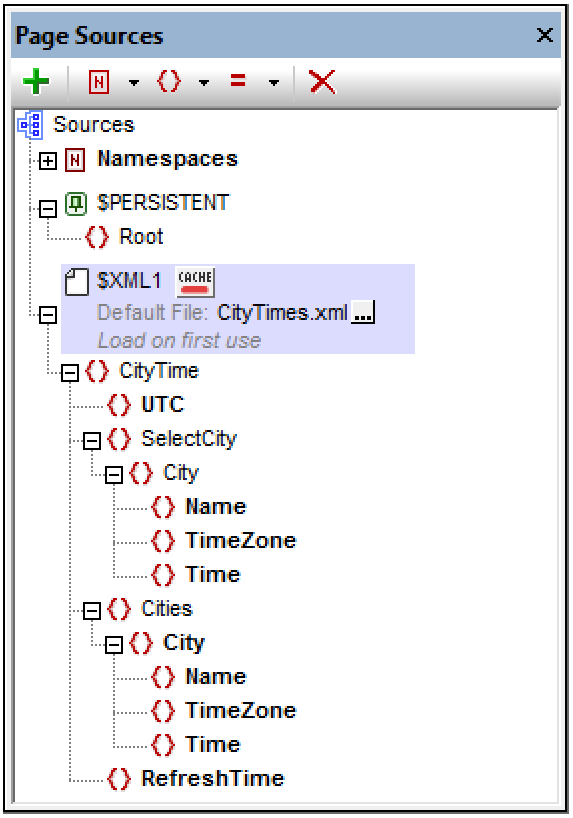
Located in the following MobileTogether folder of the *(My) Documents folder*[72]: **`MobileTogetherDesignerExamples\Tutorials\SoapRequests`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<CityTime xmlns="http://www.Nanonull.com/TimeService/">
        <UTC>12:00 AM</UTC>
        <RefreshTime>60</RefreshTime>
        <SelectCity>
                <City>
                        <Name>UTC Time</Name>
                        <TimeZone>GMT</TimeZone>
                        <Time>12:00 AM</Time>
                </City>
        </SelectCity>
        <Cities>
                <City>
                        <Name>Beijing</Name>
                        <TimeZone>UTC+8</TimeZone>
                        <Time>12:00 AM</Time>
                </City>
                <City>
                        <Name>Boston</Name>
                        <TimeZone>UTC-6</TimeZone>
                        <Time>12:00 AM</Time>
                </City>
                <City>
                        <Name>London</Name>
                        <TimeZone>GMT</TimeZone>
                        <Time>12:00 AM</Time>
                </City>
                <City>
                        <Name>Los Angeles</Name>
                        <TimeZone>UTC-8</TimeZone>
                        <Time>12:00 AM</Time>
                </City>
                <City>
                        <Name>Madrid</Name>
                        <TimeZone>UTC+1</TimeZone>
                        <Time>12:00 AM</Time>
                </City>
                <City>
                        <Name>Moscow</Name>
                        <TimeZone>UTC+3</TimeZone>
                <Time>12:00 AM</Time>
            </City>
            <City>
                <Name>Paris</Name>
                <TimeZone>UTC+1</TimeZone>
                <Time>12:00 AM</Time>
            </City>
            <City>
                <Name>Sydney</Name>
                <TimeZone>UTC+11</TimeZone>
                <Time>12:00 AM</Time>
            </City>
            <City>
                <Name>Tokyo</Name>
                <TimeZone>UTC+9</TimeZone>
                <Time>12:00 AM</Time>
            </City>
            <City>
                <Name>Vienna</Name>
                <TimeZone>UTC+1</TimeZone>
                <Time>12:00 AM</Time>
            </City>
        </Cities>
    </CityTime>
```

## Adding the XML page source

An XML page source (`$XML1`) that uses `CityTimes.xml` as its page source has been added to the design's page sources[315]. The page sources was added as follows:

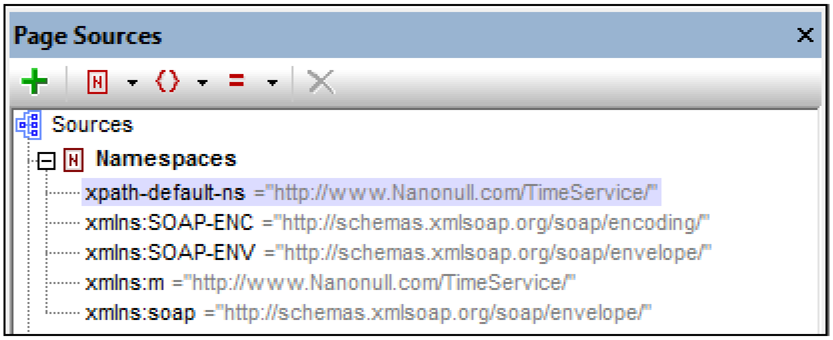
1. Click the **Add Page Source** icon in the Page Sources Pane[270] (*see screenshot above*). Select *New XML, HTML or JSON structure imported from file*. Confirm the page source settings in the next screen (without changing the defaults) by clicking **Finish**.



2. An Open dialog appears. Browse for the `CityTimes.xml` file (*see location*[215]), and click **Open**. The `$XML1` tree is created. It has the `CityTimes.xml` file set as its default file, and its XML tree has the structure of the default file.

## Namespaces of nodes in the SOAP response and the XML tree

Nodes in the SOAP response from this particular web service are unprefixed and belong to the namespace: `http://www.Nanonull.com/TimeService/`. Therefore, one way to correctly target SOAP response nodes that are entered in the design's XPath expressions is to set the XPath default namespace to this namespace[352] (*screenshot below*).



xpath-default-ns ="http://www.Nanonull.com/TimeService/"
xmlns:SOAP-ENC ="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV ="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:m ="http://www.Nanonull.com/TimeService/"
xmlns:soap ="http://schemas.xmlsoap.org/soap/envelope/"

Setting this namespace as the XPath default namespace means that **all unprefixed** nodes in the design's XPath expressions will be considered to be in this namespace. Now, if the nodes of the XML tree are also unprefixed (as is the case with our XML tree), and if these tree nodes are entered without prefixes in XPath expressions, then, in XPath expressions, these nodes will also be considered to be in the XPath default namespace: `http://www.Nanonull.com/TimeService/`.

Because of this we assign the namespace `http://www.Nanonull.com/TimeService/` also to the nodes XML tree. If you look at the XML listing above, you will notice that the document's root element has been assigned to the namespace: .

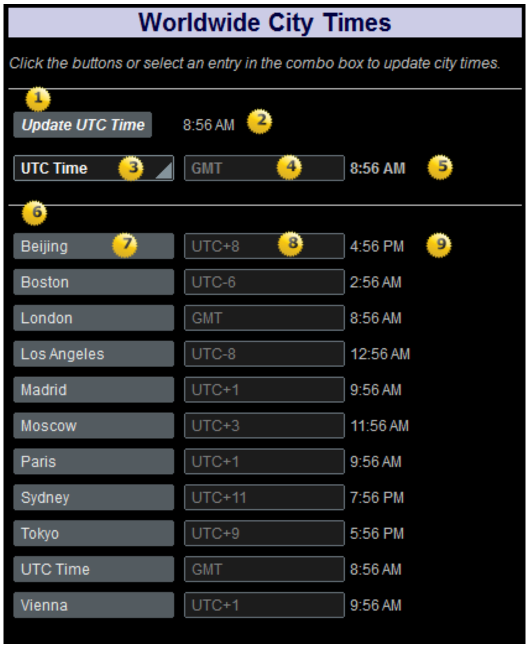
```
<CityTime xmlns="http://www.Nanonull.com/TimeService/">
```

- Since this namespace is in scope throughout the XML document and is not overridden by any namespace mapping on a descendant element, the namespace applies across the document.
- Since the document's namespace declaration has no prefix, this namespace will be the default namespace of the XML document. As a result, nodes with unprefixed local names are in this namespace.

**Note:** If your XML document is in some namespace other than the namespace of SOAP response nodes, then it is best, in the XML document, to declare the document's namespace with a prefix. Then, in the design, make sure that this same `prefix:namespace` value has been correctly entered in the design's collection of namespaces 352. In the design's XPath expressions, you should use the declared prefix when referencing XML tree nodes. Alternatively, in XPath expressions, you could address a node with the star prefix, like this **`*:NodeName`**. This would match all nodes that have the local name **`NodeName`**, no matter in what namespace the node is.

## 4.8.2 Design Components

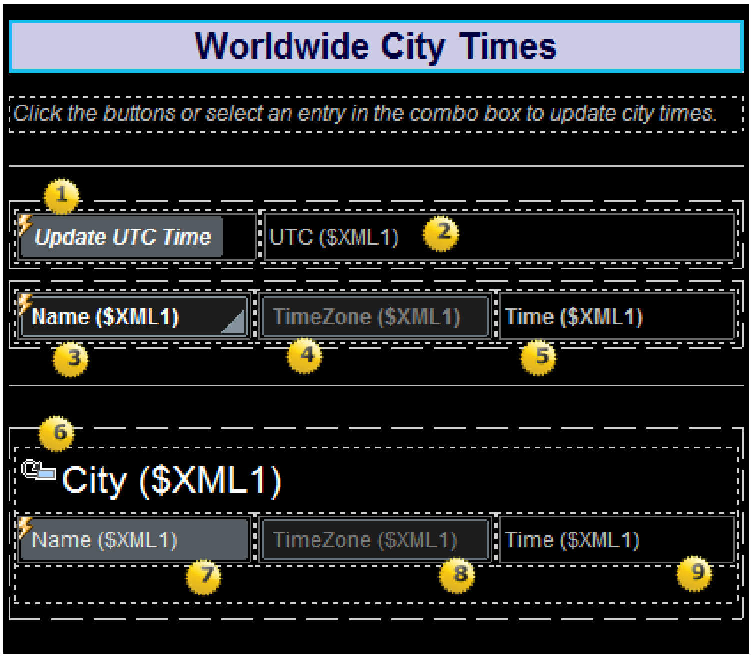
The design components are numbered in the screenshots below and are described in callouts further below. The screenshot at left shows the simulation, the screenshot at right shows the design. Click a callout to see a description of the corresponding design component.



Worldwide City Times

*Click the buttons or select an entry in the combo box to update city times.*

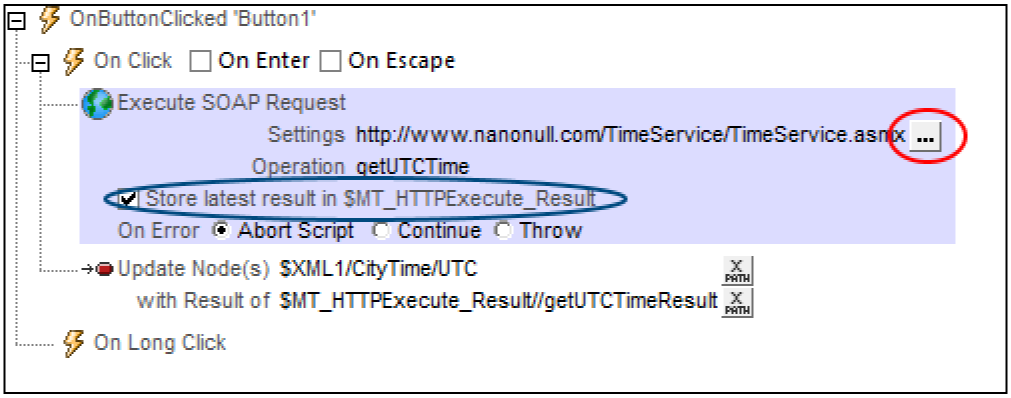
| | | |
|---|---|---|
| Update UTC Time | 8:56 AM | |
| UTC Time | GMT | 8:56 AM |
| Beijing | UTC+8 | 4:56 PM |
| Boston | UTC-6 | 2:56 AM |
| London | GMT | 8:56 AM |
| Los Angeles | UTC-8 | 12:56 AM |
| Madrid | UTC+1 | 9:56 AM |
| Moscow | UTC+3 | 11:56 AM |
| Paris | UTC+1 | 9:56 AM |
| Sydney | UTC+11 | 7:56 PM |
| Tokyo | UTC+9 | 5:56 PM |
| UTC Time | GMT | 8:56 AM |
| Vienna | UTC+1 | 9:56 AM |



*Simulation of the solution at runtime.* *The page design.*

1: Button to update UTC time

When the *Update UTC Time* button[417] is clicked at runtime, the **`OnButtonClicked`** event triggers two actions (*see screenshot below*). First, an Execute SOAP Request[835] action sends a SOAP request for the UTC time to the web service. The SOAP response from the web service is stored in the **`$MT_HTTPExecute_Result`**[1304] variable (*circled in blue in the screenshot below*). Next, an Update Node[886] action updates the `$XML1/CityTime/UTC` node with the UTC time. The content of this node is immediately displayed in a label (*see Callout 2 below*).



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx ...
Operation getUTCTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw
Update Node(s) $XML1/CityTime/UTC
with Result of $MT_HTTPExecute_Result//getUTCTimeResult
On Long Click

The web service provides an operation (**`getUTCTime`**) that gets the current UTC time. To see how the SOAP request has been defined, click the **Edit** button of the Execute SOAP Request action (*circled in red in the screenshot above*). In the SOAP Request dialog[835] that appears, the text of the SOAP request is shown in the Preview pane.

2: Label displaying UTC time

This label is associated with the page source node `$XML1/CityTime/UTC` (the label's page source link[404]). Data from this node will be displayed in the label. Since the node `$XML1/CityTime/UTC` is updated when the *Update UTC Time* button is clicked (*see Callout 1 above*), the updated UTC time is immediately displayed in this label.

3: Combo box for selecting cities

The purpose of the combo box[459] is the following:

1. To display the names of the cities that are listed in the **`Cities`** element of the XML page source[217]
2. When the user selects a city, to send a SOAP request for the current time in that city
3. To update all **`//Time`** and **`//Timezone`** nodes that are affected by the user selection (*see the 'Update actions' in the screenshot below*)

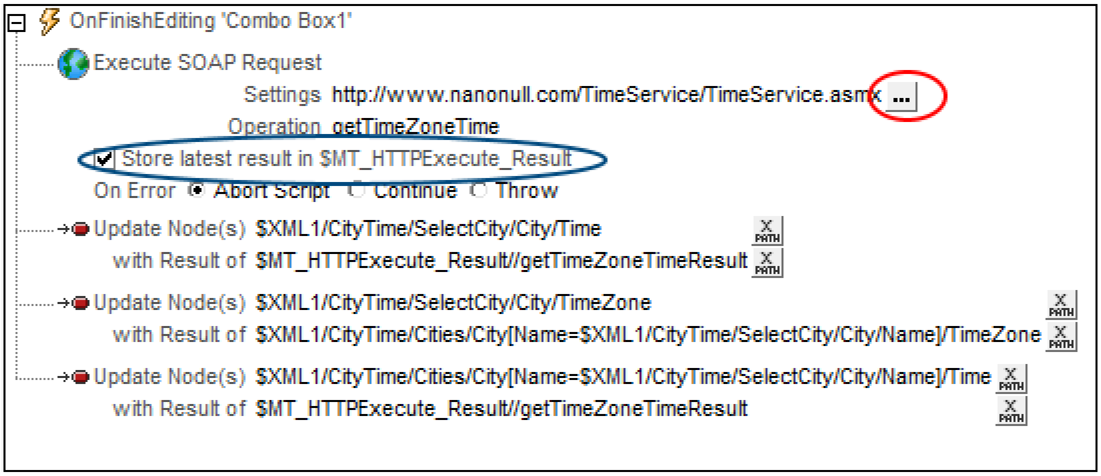
*Selecting items in the dropdown list of the combo box*
In the design, double-click the combo box control to display the Edit Combo Box dialog[459]. The items of the combo box dropdown list are the names of the cities in the **`Cities`** element of the XML page

source[217]. These city names are selected with the XPath expression `$XML1/CityTime/Cities/City/Name`. The XML value of these city names has been set to be the same as the text of the city name (the visible entry in the dropdown list of the combo box).

When the user selects a city in the combo box, the XML value of the selection (which is the same as combo box entry) is passed to the node `$XML1/CityTime/SelectCity/City/Name`. This is achieved by creating a page source link[404] between the combo box and this XML tree node (and done by dragging-and-dropping the tree node from the Page Sources Pane[270] onto the control).

*Defining the SOAP request to get the current time of a city*

Double-click the *Control Actions* symbol at the top left of the combo box to open the Actions dialog of the combo box (*screenshot below*). An Execute SOAP Request[835] action is defined for the **`OnFinishEditing`** event. The web service provides an operation (**`getTimeZoneTimeResult`**) that gets the current time of a specified timezone. The timezone for which the time is required is sent as a parameter of the SOAP request. To see how the SOAP request has been defined, click the **Edit** button of the Execute SOAP Request action (*circled in red in the screenshot below*). The SOAP Request dialog[835] appears.



OnFinishEditing 'Combo Box1'
Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx ...
Operation getTimeZoneTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw
Update Node(s) $XML1/CityTime/SelectCity/City/Time
with Result of $MT_HTTPExecute_Result//getTimeZoneTimeResult
Update Node(s) $XML1/CityTime/SelectCity/City/TimeZone
with Result of $XML1/CityTime/Cities/City[Name=$XML1/CityTime/SelectCity/City/Name]/TimeZone
Update Node(s) $XML1/CityTime/Cities/City[Name=$XML1/CityTime/SelectCity/City/Name]/Time
with Result of $MT_HTTPExecute_Result//getTimeZoneTimeResult

In the SOAP Request dialog[835], the text of the SOAP request is shown in the Preview pane, the timezone parameter is shown in the Parameters pane. Click the parameter's **XPath** button to see the XPath expression that selects the value of the **`m:timezone`** parameter:

```
for $i in $XML1/CityTime/SelectCity/City/Name return
$XML1//Cities/City/TimeZone[../Name=$i]
```

The XPath expression first selects the name of the city that the user has selected in the combo box, and stores this value in the expression's `$i` variable. The expression then goes on to select (from the **`Cities`** element of the XML page source[217]) the **`Timezone`** element of the city that has a **`Name`** element that matches the value in `$i`. In this way, the timezone of the user-selected city is set as the **`m:timezone`** parameter of the SOAP request. On receiving this request, the web service will return the current time of the requested timezone.

*Storing the SOAP response in a variable*

The SOAP response from the web service is stored in the **$MT_HTTPExecute_Result**[1304] variable (*circled in blue in the screenshot above*). Note that the entire SOAP response, which is an XML document, is stored in the variable. You will need to know the structure of the SOAP response in order to select the node containing the timezone time. In our case, the following XPath expression locates the timezone time in the stored SOAP response:

```
$MT_HTTPExecute_Result//getTimeZoneTimeResult
```

**Note:** The **getTimeZoneTimeResult** node is unprefixed in the SOAP response and it is in the **http://www.Nanonull.com/TimeService/** namespace. The design's XPath default namespace was therefore changed to this namespace[220]. If this is not done, the timezone time in the SOAP response can alternatively be accessed with the following XPath expression: `$MT_HTTPExecute_Result//*:getTimeZoneTimeResult`, which looks for the element node `getTimeZoneTimeResult` in any namespace. *See also: Namespaces of Nodes in the SOAP Response and the XML Tree*[220].

*Updating nodes with the timezone time*
The Update Node[886] action is used to update two XML tree[218] nodes with the received timezone time: (i) **$XML1/CityTime/SelectCity/City/Time**, and (ii) **$XML1/CityTime/Cities/City[Name=$XML1/CityTime/SelectCity/City/Name]/Time**. The highlighted part in the second expression specifies that only that city in the **Cities** database should be updated that has a name that matches the name of the user-selected city. The content of these updated nodes are immediately displayed in labels via page source links[404] (*see Callouts 5 and 9 below*). The value of the timezone time is obtained from the SOAP response via the **$MT_HTTPExecute_Result**[1304] variable.

*Displaying the timezone of the selected city*
The Update Node[886] action is used to update the `$XML1/CityTime/SelectCity/City/TimeZone` node. The value with which the node is updated is the content of the node selected with the expression: **$XML1/CityTime/Cities/City[Name=$XML1/CityTime/SelectCity/City/Name]/Timezone**. This expression selects the **TimeZone** element of that city in the **Cities** database that has a name that matches the name of the user-selected city. The content of the updated node is immediately displayed in an edit field via a page source link[404] (*see Callout 4 below*).

▼ 4: Edit field displaying the timezone of the user-selected city

This edit field is associated with the page source node `$XML1/CityTime/SelectCity/City/TimeZone` (the edit field's page source link[404]). So, as soon as the user selects a city in the combo box, that city's timezone is displayed in the edit field. The chain of actions is as follows: When the user selects a city, the **SelectCity//TimeZone** node is updated (because of the *Update* action of the combo box; *see Callout 3 above*). Then, because the **SelectCity//TimeZone** node is the page source link of the edit field, the edit field automatically displays the updated value of the **SelectCity//TimeZone** node.

▼ 5: Label displaying the current time of the user-selected city

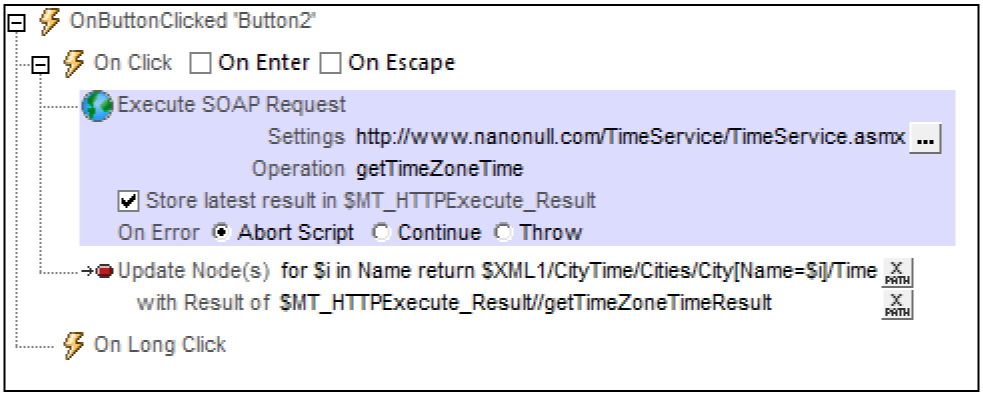
This label is associated with the page source node `$XML1/CityTime/SelectCity/City/Time` (the label's page source link[404]). When the user selects a city in the combo box, (i) a SOAP request is sent for the current time in the timezone of that city, and (ii) the **`SelectCity//Time`** node is updated with the current time in that timezone (because of the *Update* action of the combo box; *see Callout 3 above*). The label then automatically displays the updated time because of the page source link to the updated **`SelectCity//Time`** node.

6: Table displaying the Cities database

The cities in the **`Cities`** element of the XML page source[218] are each defined in a **`City`** element. The **`City`** element has therefore been created as the repeating row in a table with three columns and dynamic rows. Each city is displayed in a row. The columns display, respectively, each city's name, timezone, and time. The controls used in the columns are, respectively, a button (with the city's **`Name`** element as its page source link[404]), an edit field (with the city's **`Timezone`** element as its page source link), and a label (with the city's **`Time`** element as its page source link). *See Callouts 7,8 and 9 below.*

7: Button to update a city's time

The button displays the name of the city by way of a page source link to the `$XML1/CityTime/Cities/City/Name`. At runtime, when the button of a city is clicked, then a SOAP request is sent to get that city's (timezone) time (*see screenshot below*). The value of the **`m:timezone`** parameter of the request is obtained from the city's **`TimeZone`** element. Since the context node is **`City`**, the XPath expression to fetch the city's timezone will be: **`TimeZone`**. The SOAP response is stored in the **`$MT_HTTPExecute_Result`**[1304] variable. Next, an Update Node[886] action updates the `$XML1/CityTime/Cities/City/Name` node with the timezone time. The content of this updated node is immediately displayed in a label (*see Callout 9 below*).


OnButtonClicked 'Button2'
On Click □ On Enter □ On Escape
Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx ...
Operation getTimeZoneTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw
Update Node(s) for $i in Name return $XML1/CityTime/Cities/City[Name=$i]/Time
with Result of $MT_HTTPExecute_Result//getTimeZoneTimeResult
On Long Click

8: Edit field displaying the timezone of cities in the database

The edit field is associated with the XML node `$XML1/CityTime/Cities/City/TimeZone` (the edit field's page source link[404]). The content of this node does not change[218].

9: Label displaying a city's current time

The label is associated with the XML node `$XML1/CityTime/Cities/City/Time` (the label's page source link[404]). Data from this node is displayed in the label as soon as the user clicks the corresponding **`City`** button (*Callout 7 above*). This is because (i) the button has an action to update this node (*see Callout 7 above*), and (ii) this node is the label's page source link[404].

## Page actions

To view the page actions, right-click inside the page and select **Page Actions**. In the dialog that appears, three actions have been defined for the `OnPageLoad` event. These actions will be executed when the page loads. They provide data for the initial page display.

The following actions have been defined:

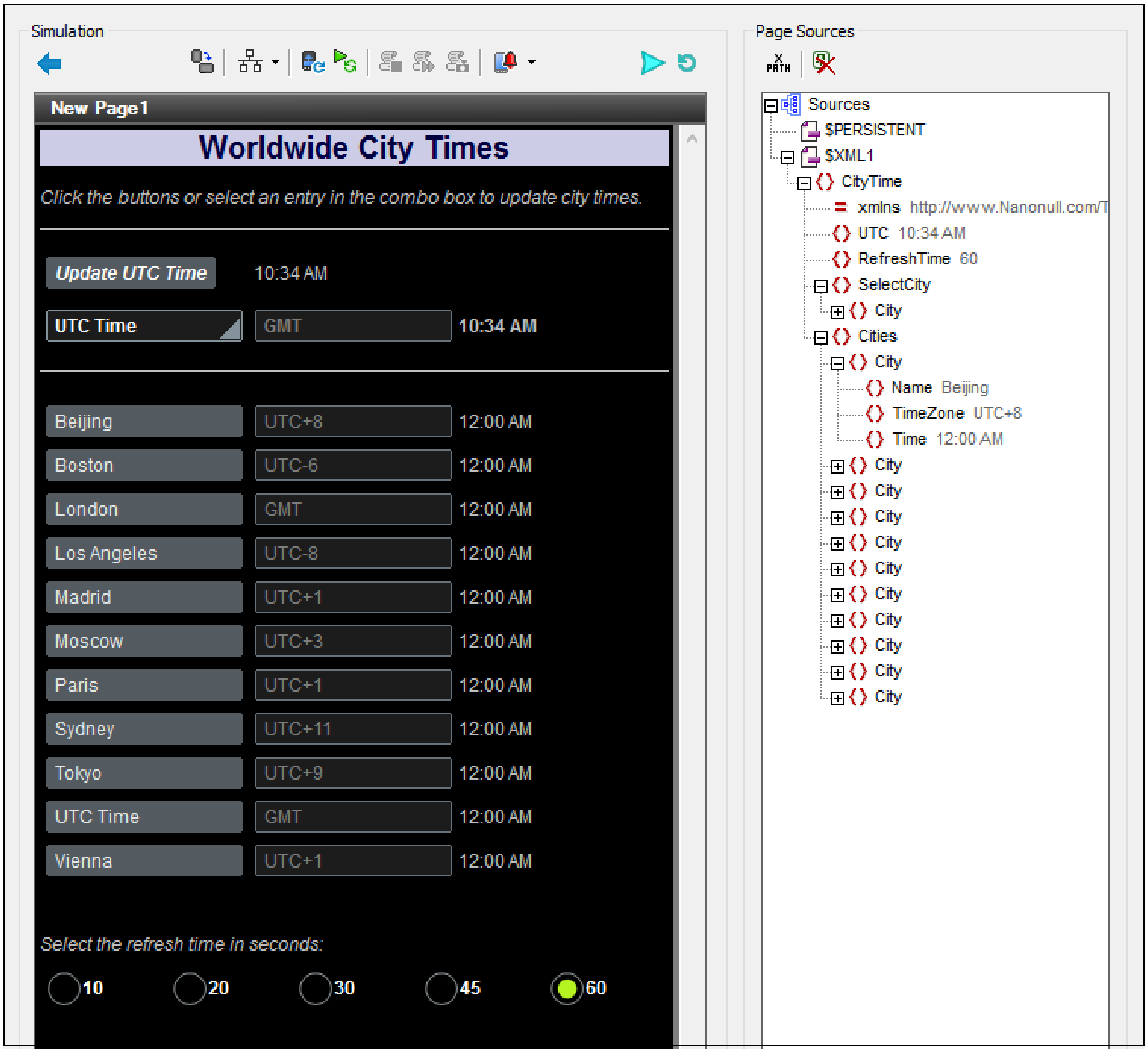
- Execute SOAP Request[835]: The action requests the UTC time from the web service and stores the response in the **`$MT_HTTPExecute_Result`**[1304] variable. The request is defined in the same way as for the UTC Time button (*Callout 1*).
- Update Node[886]: Updates the node `$XML1/CityTime/UTC` with the UTC time. Since this node is the page source link[404] of the UTC Time label (*Callout 2*), the label will be initialized with the current UTC time.
- Update Node[886]: Updates the node `$XML1/CityTime/SelectCity/City/Time` with the UTC time. Since the initial value of the selected city (**`SelectCity//Name`**) is `UTC Time` (*see the XML file*[218]), we initialize the **`SelectCity//Time`** node with the current UTC time.

## 4.8.3 Refreshing the Page

The design components that we have described till now[221] execute a SOAP request when the user triggers an event (either by clicking a button or selecting a city in the combo box). This means that the UTC time and city times in the display will be incorrect very soon if the user does not update them manually. However, there is a way to continually update the times in the display. This is by using the `OnPageRefresh`[390] event.

To automatically refresh the display of the city times, we have used the following mechanism.

- A node in the XML file[217] has been defined to hold the amount of time, in seconds, between refreshes: `$XML1/CityTime/RefreshTime`
- The `OnPageRefresh`[390] event has been defined such that the page refreshes: (i) when the user taps the device's **Refresh** button (*see screenshot below*), and (ii) after every `X` number of seconds, where `X` is the number in the `$XML1/CityTime/RefreshTime` node. For each page refresh, a set of actions has been defined that updates the city times in the display
- A set of radio buttons allows the user to select the page refresh interval to be `10/20/30/45/60` seconds.



Worldwide City Times

*Click the buttons or select an entry in the combo box to update city times.*

| | | |
|---|---|---|
| Update UTC Time | 10:34 AM | |
| UTC Time | GMT | 10:34 AM |
| Beijing | UTC+8 | 12:00 AM |
| Boston | UTC-6 | 12:00 AM |
| London | GMT | 12:00 AM |
| Los Angeles | UTC-8 | 12:00 AM |
| Madrid | UTC+1 | 12:00 AM |
| Moscow | UTC+3 | 12:00 AM |
| Paris | UTC+1 | 12:00 AM |
| Sydney | UTC+11 | 12:00 AM |
| Tokyo | UTC+9 | 12:00 AM |
| UTC Time | GMT | 12:00 AM |
| Vienna | UTC+1 | 12:00 AM |

*Select the refresh time in seconds:*

○ 10 ○ 20 ○ 30 ○ 45 ● 60

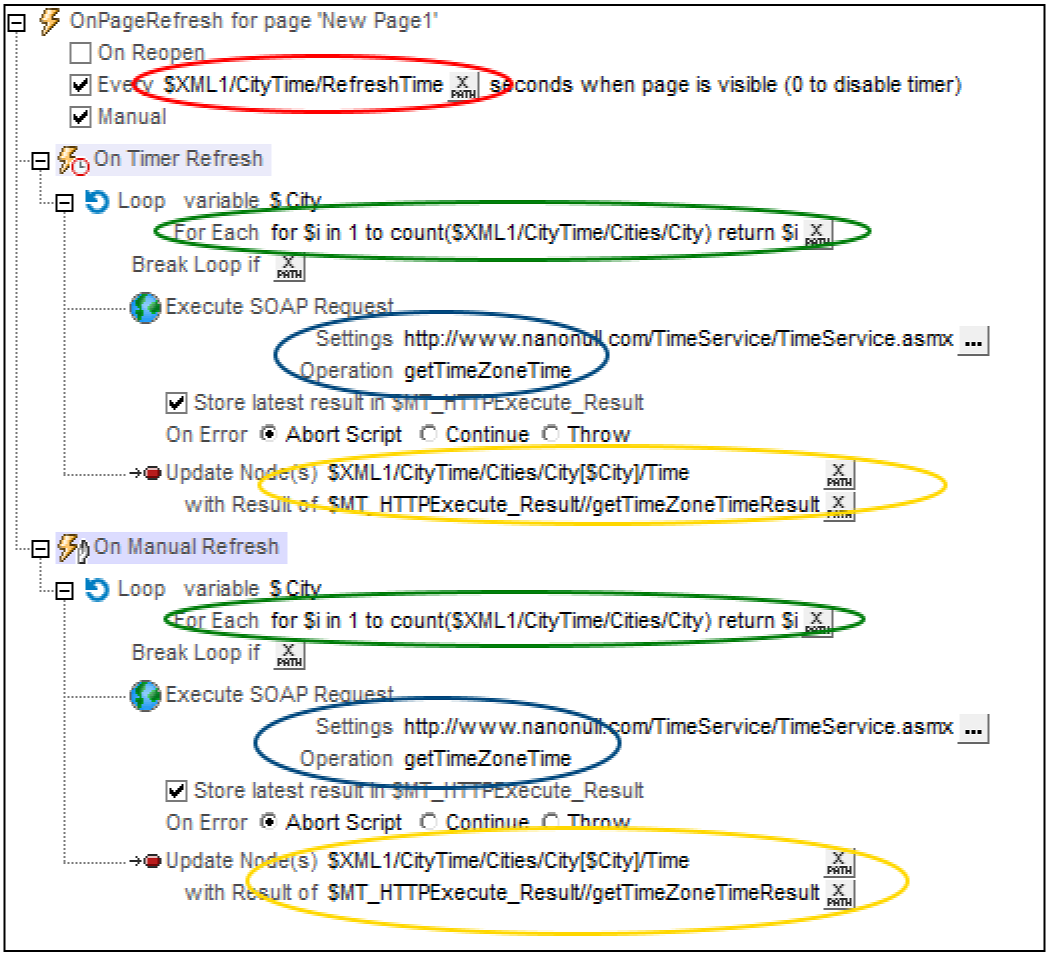
```
Sources
  $PERSISTENT
  $XML1
    CityTime
      xmlns http://www.Nanonull.com/T
      UTC 10:34 AM
      RefreshTime 60
      SelectCity
        City
      Cities
        City
          Name Beijing
          TimeZone UTC+8
          Time 12:00 AM
        City
        City
        City
        City
        City
        City
        City
        City
        City
        City
```

The key points of this mechanism are described below.

## The OnPageRefresh event and its actions

Three methods are available to define when a page is refreshed (*see screenshot below*). We have chosen the following two methods to refresh the page:

- On a timer, every `X` number of seconds. The number of seconds has been defined as the content of the `$XML1/CityTime/RefreshTime` element (*definition is circled in red in screenshot below*).
- Manually, when the user taps the device's **Refresh** button (*see screenshot above*).



```
OnPageRefresh for page 'New Page1'
  ☐ On Reopen
  ☑ Every $XML1/CityTime/RefreshTime seconds when page is visible (0 to disable timer)
  ☑ Manual
  On Timer Refresh
    Loop  variable $City
      For Each  for $i in 1 to count($XML1/CityTime/Cities/City) return $i
      Break Loop if
      Execute SOAP Request
          Settings  http://www.nanonull.com/TimeService/TimeService.asmx ...
          Operation  getTimeZoneTime
        ☑ Store latest result in $MT_HTTPExecute_Result
        On Error  ● Abort Script  ○ Continue  ○ Throw
      Update Node(s)  $XML1/CityTime/Cities/City[$City]/Time
        with Result of  $MT_HTTPExecute_Result//getTimeZoneTimeResult
  On Manual Refresh
    Loop  variable $City
      For Each  for $i in 1 to count($XML1/CityTime/Cities/City) return $i
      Break Loop if
      Execute SOAP Request
          Settings  http://www.nanonull.com/TimeService/TimeService.asmx ...
          Operation  getTimeZoneTime
        ☑ Store latest result in $MT_HTTPExecute_Result
        On Error  ● Abort Script  ○ Continue  ○ Throw
      Update Node(s)  $XML1/CityTime/Cities/City[$City]/Time
        with Result of  $MT_HTTPExecute_Result//getTimeZoneTimeResult
```

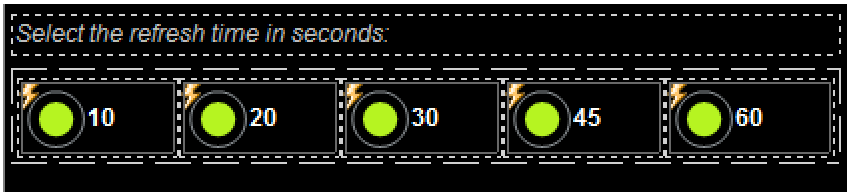
Since we wish to update the current time of each city in the display, we have done the following:

- Created a loop[897] that iterates over each city (*definition circled in green*). The loop returns a sequence of integers. Each integer is tied to a **`City`** node by being the index of that **`City`** node. We do not wish to iterate directly over the City nodes because we wish to update these nodes within the loop, and updates are not possible while the nodes are being iterated over[897].
- Within the loop, that is, for each city, (i) executed a SOAP request to get the **`TimeZoneTime`** of that city (*circled in blue*), and (ii) updated that city's **`Time`** node with the current time in that city's timezone (*circled in yellow*).

The Loop action[897] is the same for both types of refresh, and it updates the current time of each city in the database.

## Enabling the user to select the refresh interval

To enable the user to select the interval at which the page refreshes (and updates the city times), we have created a set of five radio buttons[571] (*see screenshot below*).



The radio buttons have the following settings:

- A **`Text`** property value and a **`Checked Values`** property value, both set to the refresh interval in seconds: `10/20/30/45/60` seconds. The **`Text`** property value display the value near the radio button (*see screenshot above*). The **`Checked Values`** property value is the XML data value that will be used if the radio button is selected.
- All five radio buttons have a page source link[404] to the `$XML1/CityTime/RefreshTime` element. This means that they form a mutually exclusive set, and that the **`Checked Value`** of the selected radio button will become the content of the `RefreshTime` element.
- Each button has a Restart/Stop Page Timer[791] action defined for its **`OnFinishEditing`** event. This is required in order to restart the page timer (defined in the **`OnPageRefresh`** event; *see above*) with the new refresh interval. Remember that the timer takes its refresh interval from the `$XML1/CityTime/RefreshTime` element (*see above*), and that the radio button selection has just updated this element node (because of the page source link to the node).

## Running a simulation to test the page refresh

Press **F5** to run a simulation in MobileTogether Designer[1356]. The timer for the page-refresh will be started with a value that is taken from the `$XML1/CityTime/RefreshTime` node. This is 60 (seconds) in the original data tree.

- When you select one of the radio buttons, the **`Checked Value`** of that button is passed to the `$XML1/CityTime/RefreshTime` node, and the timer is restarted (defined with the button event's Restart/Stop Page Timer[791] action) with the user-selected page-refresh time.
- You can also click, at any time, the **Refresh** button at the top right of the simulator (*see first screenshot of this topic*) to manually refresh the display of city times.

# 4.9 Sharing Geolocations

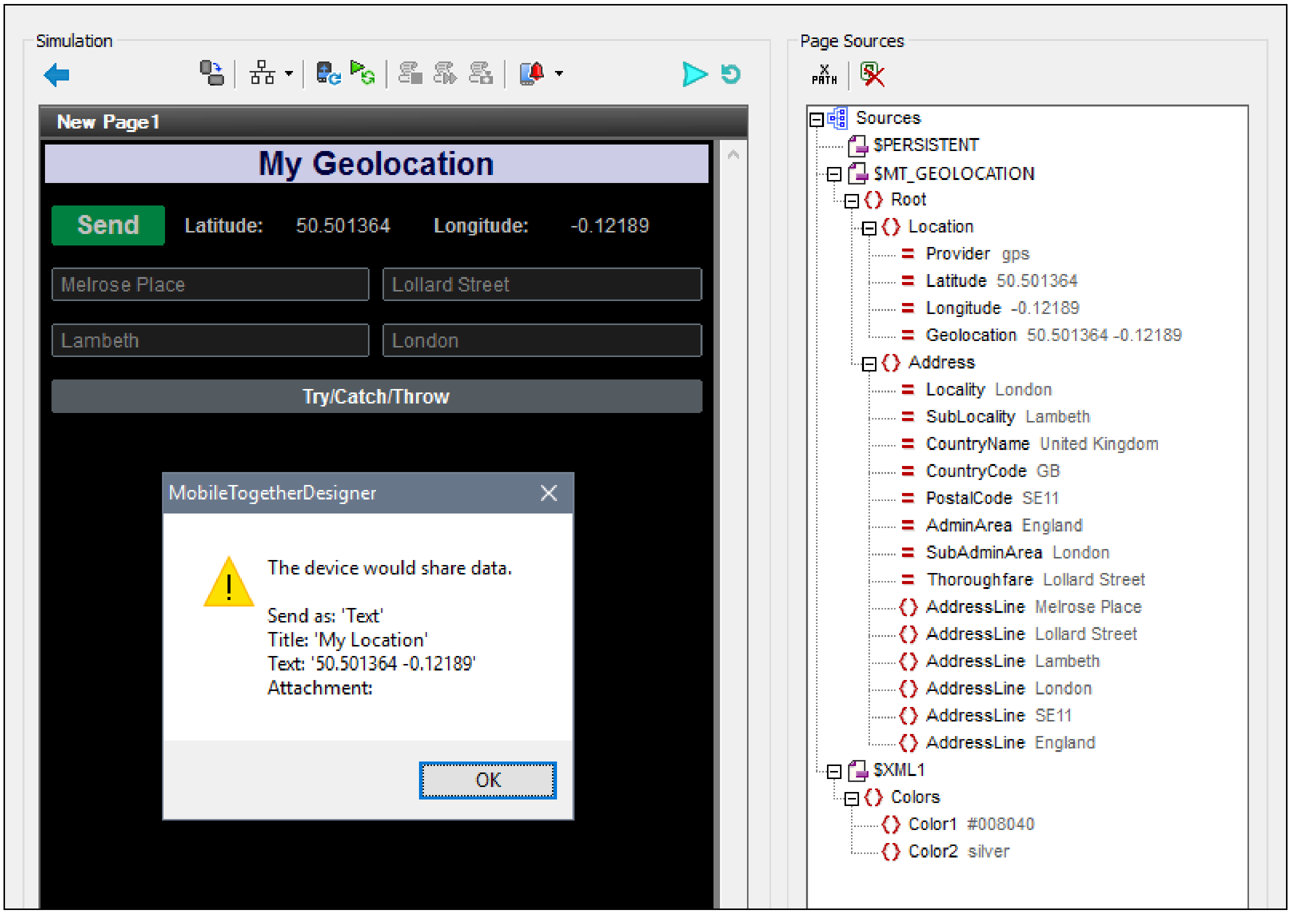
This tutorial shows how to do the following:

- Read the mobile device's current geolocation data and write this data into the design's `$MT_GEOLOCATION` tree
- Access the `$MT_GEOLOCATION` tree in order to display the geolocation data in the mobile device
- Share the geolocation data with contacts via the device's messaging and social networking apps
- Throw exceptions when errors occur, and display these exceptions

## What the solution does and displays

The screenshot below shows a simulation of the design in MobileTogether Designer. The design's functionality is accessed via two buttons:

- **Send:** Starts tracking the device's geolocation, writes the geolocation data into the `$MT_GEOLOCATION` tree, displays key geolocation items in the solution, and opens the mobile device's Share menu.
- **Try/Catch/Throw:** Displays a warning message if the geolocation coordinates are outside the USA.



Latitude: 50.501364 Longitude: -0.12189

Melrose Place | Lollard Street

Lambeth | London

The device would share data.

Send as: 'Text'
Title: 'My Location'
Text: '50.501364 -0.12189'
Attachment:

```
Sources
  $PERSISTENT
  $MT_GEOLOCATION
    Root
      Location
        Provider gps
        Latitude 50.501364
        Longitude -0.12189
        Geolocation 50.501364 -0.12189
      Address
        Locality London
        SubLocality Lambeth
        CountryName United Kingdom
        CountryCode GB
        PostalCode SE11
        AdminArea England
        SubAdminArea London
        Thoroughfare Lollard Street
        AddressLine Melrose Place
        AddressLine Lollard Street
        AddressLine Lambeth
        AddressLine London
        AddressLine SE11
        AddressLine England
  $XML1
    Colors
      Color1 #008040
      Color2 silver
```

## The tutorial files

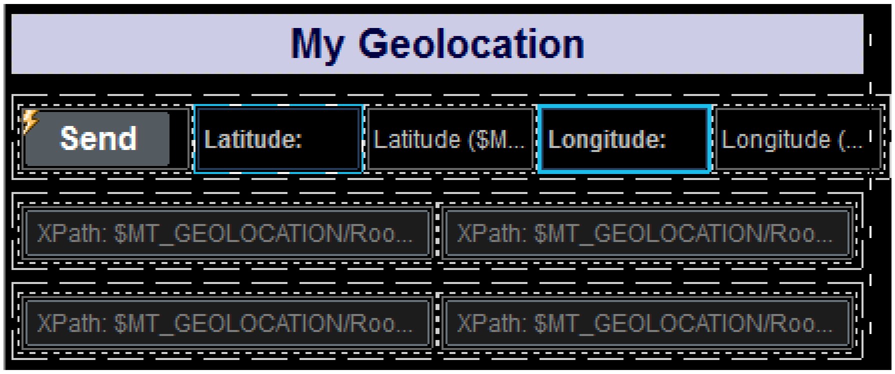
The files for this tutorial are located in your (72 My) Documents 72 MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\Geolocations**.

- SharingGeolocations.mtd: This is the completed MobileTogether design file. Open this file and refer to it while reading this tutorial. You can run a simulation in MobileTogether Designer by pressing **F5**.
- LondonLocations.xml: This is an XML data file that contains the geolocation data of a London location. Since the simulation is done on a desktop, we use the data from this file to stand in for the geolocation data of a mobile device.

The paths in the design file are relative, and the XML data file has not been deployed to a server. So, if you copy these two files to any folder, you will be able to correctly run simulations in MobileTogether Designer.

## 4.9.1 Reading and Sharing the Geolocation

The top part of the design (*screenshot below*) displays the geolocation data of the device.

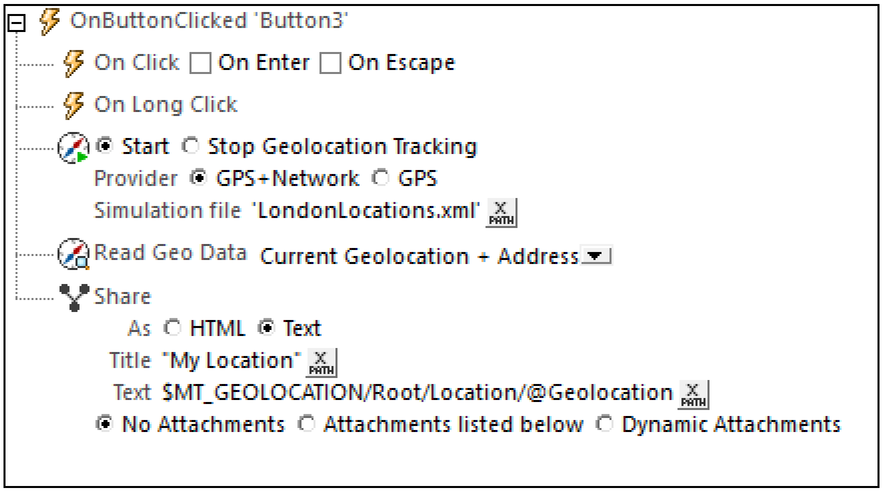

My Geolocation
Send
Latitude:
Latitude ($M...
Longitude:
Longitude (...
XPath: $MT_GEOLOCATION/Roo...
XPath: $MT_GEOLOCATION/Roo...
XPath: $MT_GEOLOCATION/Roo...
XPath: $MT_GEOLOCATION/Roo...

- The top row consists of the **Send** button and four labels 554. The two labels highlighted in blue use static text for their label text (Latitude: and Longitude: ). The other two labels have page source links 404 to the latitude and longitude nodes of the $MT_GEOLOCATION tree: **$MT_GEOLOCATION/Root/Location/@Latitude** and **$MT_GEOLOCATION/Root/Location/@Longitude**. As a result, whenever, these nodes are updated, the two labels will also be updated.
- The second and third rows contain a total of four Edit Fields 495. These controls are page source links 404, respectively, to the first four **AddressLine** nodes of the $MT_GEOLOCATION tree: **$MT_GEOLOCATION/Root/Address/AddressLine**. All four edit fields will therefore also be updated when the corresponding nodes are updated.
- This leaves the **Send** button. The button's OnButtonClicked action defines all the actions required to: (i) obtain and display the geolocation of the mobile device, and (ii) to share the geolocation via the device's apps. The **Send** button's actions are described below.

**Note:** The $MT_GEOLOCATION tree is automatically added as a page source of the page when the Start/Stop Geo Tracking 735 action or the Read Geo Data 736 action is added to design.

## Send-button actions

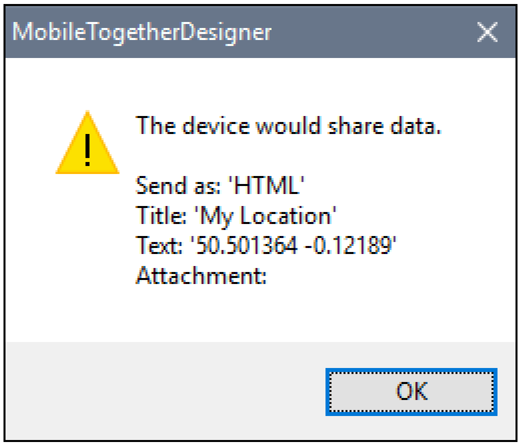
Double-click the Actions dialog icon at the top left of the **Send** button (*see screenshot above*) to open the button's Actions dialog (*screenshot below*).



OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
● Start ○ Stop Geolocation Tracking
Provider ● GPS+Network ○ GPS
Simulation file 'LondonLocations.xml'
Read Geo Data Current Geolocation + Address
Share
As ○ HTML ● Text
Title "My Location"
Text $MT_GEOLOCATION/Root/Location/@Geolocation
● No Attachments ○ Attachments listed below ○ Dynamic Attachments

The following actions have been defined for the **OnButtonClicked** event. This means that when the button is clicked, all the actions defined for it will be executed, one after the other, in the order defined.

- Start Geolocation Tracking 735 to start the tracking of the device. We have defined an XML file to be used as a simulation file. This file is called `LondonLocations.xml`, and it should be located in the same folder as the MTD file 230. Since actual geolocation data is not available during simulations on the desktop running MobileTogether Designer, the data in this file is used as a substitute for the actual geolocation data of a mobile device.
- The Read Geo Data 736 action takes the geolocation data provided by the tracking and structures it into the XML format of the `$MT_GEOLOCATION` tree. In our definition of the action, we have specified that data from both the **`Location`** element as well as the **`Address`** element should be written to the `$MT_GEOLOCATION` tree. In real life situations, this data would be the **`Location`** and **`Address`** data of the mobile device. In the case of our simulations, the **`Location`** and **`Address`** data is taken from the file `LondonLocations.xml`. When the `$MT_GEOLOCATION` tree is updated with the new geolocation data, all the labels and edit fields in the solution will automatically display the updated data. This is because of the page source links 404 between the controls and the updated nodes.
- The Share action 699 creates a text message with a title of *My Location*. the content of the message is the value of the **`$MT_GEOLOCATION/Root/Location/@Geolocation`** attribute (which is a concatenation of the latitude and longitude values). In real life execution, the Share action will open the mobile device's Share menu, thus enabling the end user to send the device's current geolocation to contacts via any of the device's Share apps. In the simulation, a message box containing the title and content of the messages is displayed (*see screenshot below*).



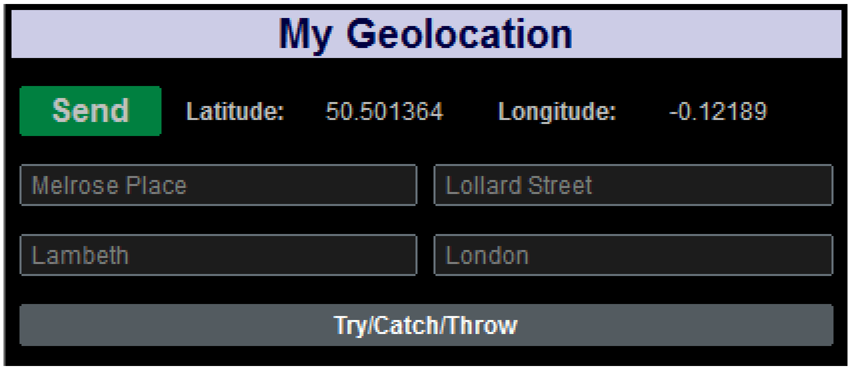
The device would share data.

Send as: 'HTML'
Title: 'My Location'
Text: '50.501364 -0.12189'
Attachment:

For more detailed information, see the description of the respective actions.

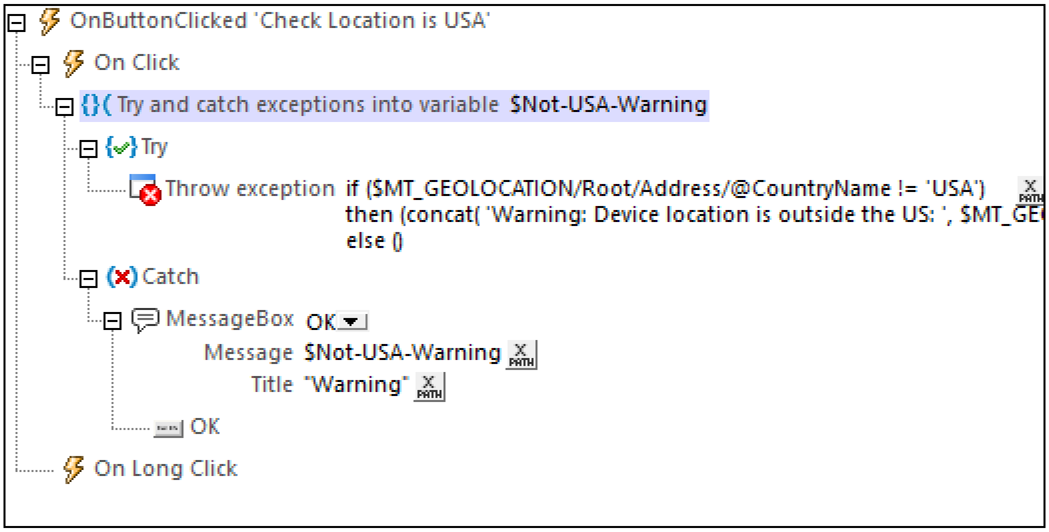
## 4.9.2 Using Try/Catch/Throw Exceptions

We have used the Try/Catch[907] and Throw[905] actions to display a warning if the geolocation coordinates describe a location outside the USA. These actions are executed when the **Try/Catch/Throw** button is clicked (*see screenshot below*).



### The Try/Catch/Throw actions

In the design, double-click the Actions dialog icon at the top left of the **Try/Catch/Throw** button to open the button's Actions dialog (*screenshot below*).



```
OnButtonClicked 'Check Location is USA'
  On Click
    Try and catch exceptions into variable  $Not-USA-Warning
      Try
        Throw exception  if ($MT_GEOLOCATION/Root/Address/@CountryName != 'USA')
                         then (concat( 'Warning: Device location is outside the US: ', $MT_GE
                         else ()
      Catch
        MessageBox  OK
              Message  $Not-USA-Warning
                Title  "Warning"
          OK
  On Long Click
```

The actions have been defined as follows:

1. A Try/Catch action was added.
2. We set up a variable `$Not-USA-Warning` which will be used to hold the exception message.
3. The Try part sets up a condition to test whether the geolocation is not in the USA. This condition is specified in the XPath expression of a Throw action[905]. If the condition is true, then an exception is thrown. The exception message will be stored in the `$Not-USA-Warning` variable. If the condition is false, then no exception is thrown; we generate the empty sequence so that nothing is stored in the `$Not-USA-Warning` variable. A description of the XPath expression that does this is given below.
4. The Catch part of the Try/Catch action is processed only if an exception is thrown (that is, only if the condition tested above evaluates to true). In the Catch part, we display a message box that shows the content of the `$Not-USA-Warning` variable.

## XPath expression of the Throw action

The XPath expression of the Throw action is:

```
if ($MT_GEOLOCATION/Root/Address/@CountryName != 'USA')
then (concat( 'Warning: Device location is outside the US: ',
$MT_GEOLOCATION/Root/Address/@CountryName))
else ()
```

This expression works as follows:

- The `if` clause checks whether the value of the `$MT_GEOLOCATION/Root/Address/@CountryName` node is not `'USA'`.
- The `then` clause is processed if the country name **is not** `USA`. This clause generates a string.
- The `else` clause is processed if the country name **is** `USA`. It produces an empty sequence

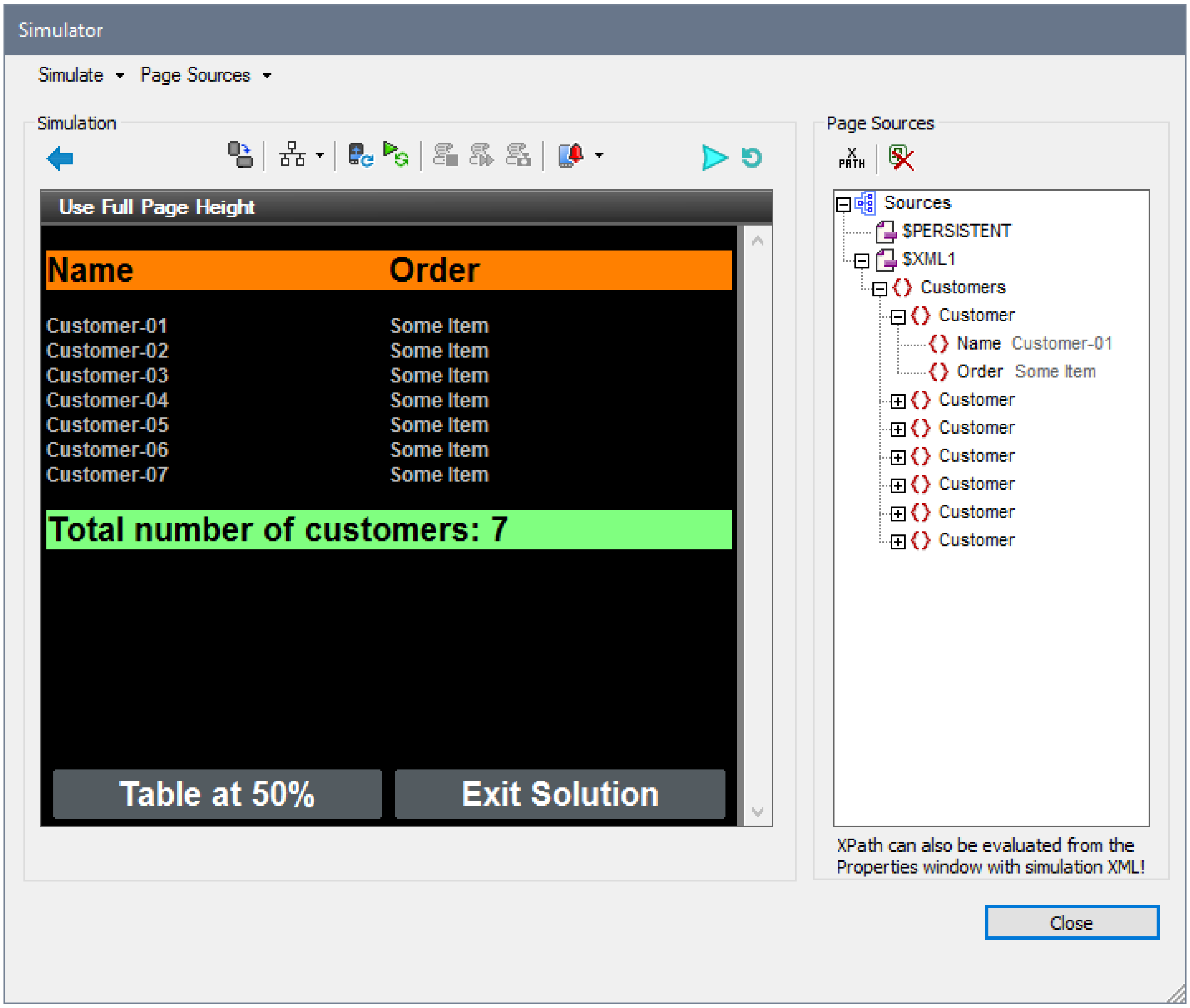
If the geolocation country **is not** `USA`, then the condition is `true` and the expression evaluates to the string generated by the `then` clause. Since this result is not an empty sequence, an exception is thrown and the generated string is stored in the Try/Catch[905] variable `$Not-USA-Warning`.

If the geolocation country **is** `USA`, then the condition is `false` and the expression evaluates to an empty sequence (generated by the `else` clause). Because the result is an empty sequence, no exception is thrown. Therefore, the Catch part of the Try/Catch action 905 is not executed.

# 4.10 Scrollable Tables

This tutorial describes the features of scrollable tables. The design file (`ScrollableTables.mtd`) contains two top pages, which show, respectively:

- The settings of a scrollable table that is used to constrain end-of-page content to the bottom of the screen (*see screenshot below*)
- A table that is displayed at a height that is a percentage of the screen height



| Name | Order |
|---|---|
| Customer-01 | Some Item |
| Customer-02 | Some Item |
| Customer-03 | Some Item |
| Customer-04 | Some Item |
| Customer-05 | Some Item |
| Customer-06 | Some Item |
| Customer-07 | Some Item |

Total number of customers: 7

XPath can also be evaluated from the Properties window with simulation XML!

## The tutorial files

The files for this tutorial are located in your (72 My) Documents 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\ScrollableTables`**.

- `ScrollableTables.mtd`: This is the completed MobileTogether design file. Open this file and refer to it while reading this tutorial. You can run a simulation in MobileTogether Designer by pressing **F5**.

- `ScrollableTables-01.xml`: This is an XML file that contains a simple and short customer database comprising seven customer records. You can see the file's structure in the **`$XML1`** tree in screenshot above.
- `ScrollableTables-02.xml`: This is a longer version of **`ScrollableTables-01.xml`**. It contains 29 records.

The paths in the design file are relative, and the XML files have not been deployed to a server. So, if you copy these three files to any folder, you will be able to correctly run the simulations in MobileTogether Designer.

### Tutorial structure

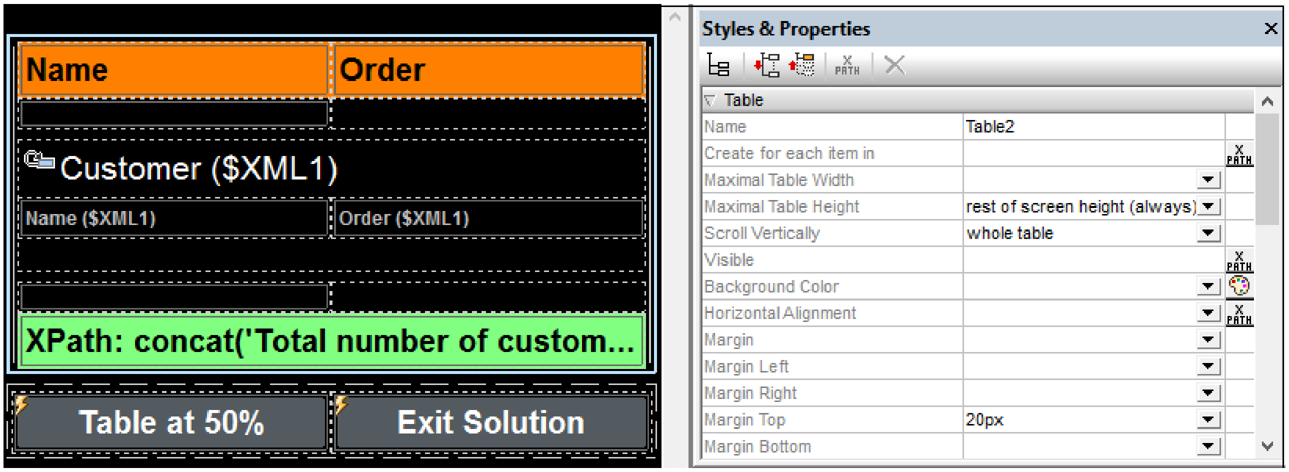
This tutorial is organized into the following sections:

- Tables that Force Full Screen Height [237] describes settings to auto-adjust table heights so that the contents of the page fill the screen completely.
- Tables Having Specific Heights [238] describes how to create tables that are a specific fraction of the screen height. If the height required to display all the rows of the table is more than the table's defined height, then the table becomes scrollable.

## 4.10.1 Tables that Force Full Screen Height

The first page of the design, *Use Full Page Height*, contains a table with dynamic rows [1069] that is created with the **`$XML1/Customers/Customer`** element as its repeating element (*see screenshot below*). This means that each **`Customer`** element is created as a Table Row Group and is displayed in its own row. The table is created with a header (styled with an orange background color in the design; *see screenshot below*) and a footer (green background color). The XML data for the table is taken from the XML file `ScrollableTables-01.xml`.

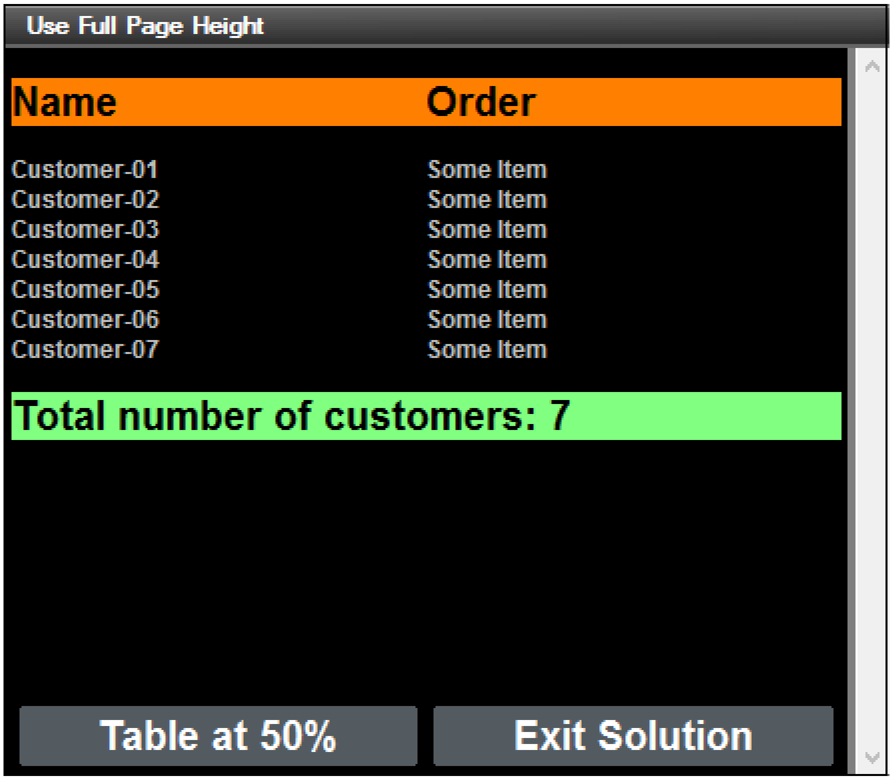
Below the table, we have created two buttons: one to go to the next page, the other to exit the solution.



| Property | Value |
|---|---|
| Name | Table2 |
| Create for each item in | |
| Maximal Table Width | |
| Maximal Table Height | rest of screen height (always) |
| Scroll Vertically | whole table |
| Visible | |
| Background Color | |
| Horizontal Alignment | |
| Margin | |
| Margin Left | |
| Margin Right | |
| Margin Top | 20px |
| Margin Bottom | |

We wish to create the design so that the two buttons always appear at the bottom of the screen, no matter what the height of the table, that is, even if the table does not have enough rows to extend to the bottom of the screen. We do this by using the following settings:

- The table's `Max Table Height` property is set to *Rest of screen height (always)*. This ensures that additional space is added below the table so that the last component of the page is displayed just above the bottom of the screen.
- The table's `Scroll Vertically` property is set to *Whole table*. This ensures that the footer is kept with the body of the table (*see screenshot below*). Otherwise, the footer would be positioned just above the rest of the page's content, which might lead to an unsightly gap between the last row of the table and the footer.


Use Full Page Height

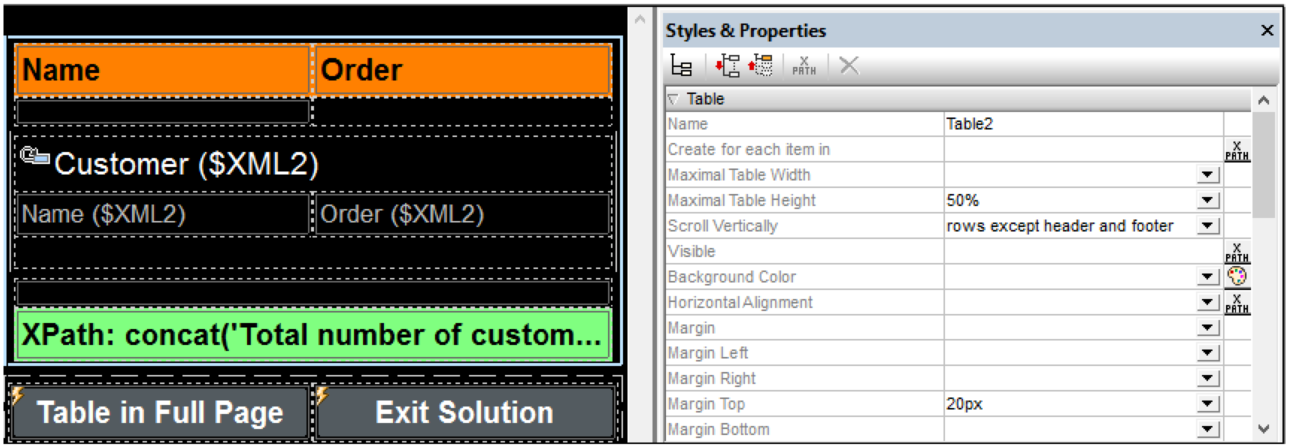
| Name | Order |
|---|---|
| Customer-01 | Some Item |
| Customer-02 | Some Item |
| Customer-03 | Some Item |
| Customer-04 | Some Item |
| Customer-05 | Some Item |
| Customer-06 | Some Item |
| Customer-07 | Some Item |
| Total number of customers: 7 | |

The point to note is the following: If the table's `Max Table Height` property is set to *Rest of screen height (always)*, then the height of the table will auto-adjust so that the components of the page are displayed to occupy the full screen.

You can modify the values of properties to test the various possibilities. See the section Table Properties 1085 for details of scrollable table properties.

## 4.10.2 Tables Having Specific Heights

The second page of the design, *Table at 50 Percent*, contains a table with dynamic rows 1069 that is created with the **`$XML2/Customers/Customer`** element as its repeating element (*see screenshot below*). This table is similar to the table created on the previous page. The difference is that while the previous table auto-adjusted its height so that all page components filled the full screen height, this table is defined to have a height that is 50% of the screen height (*see the table's Max Table Height property in the screenshot below*).

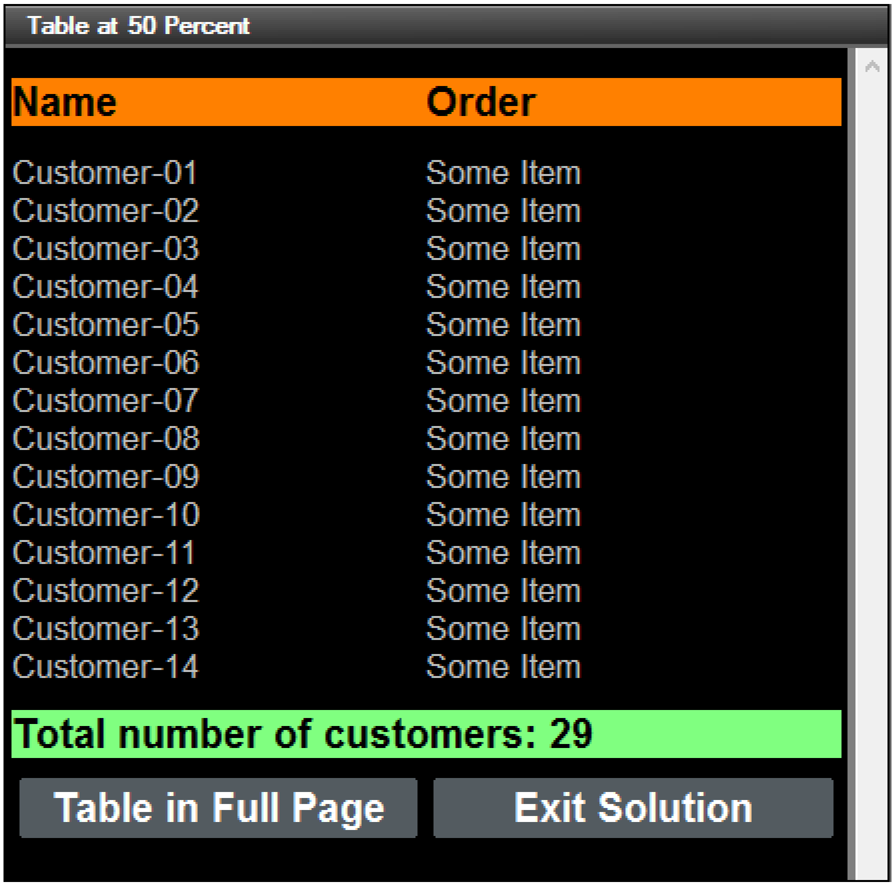


Table

| | |
|---|---|
| Name | Table2 |
| Create for each item in | |
| Maximal Table Width | |
| Maximal Table Height | 50% |
| Scroll Vertically | rows except header and footer |
| Visible | |
| Background Color | |
| Horizontal Alignment | |
| Margin | |
| Margin Left | |
| Margin Right | |
| Margin Top | 20px |
| Margin Bottom | |

As a result of the *50%* `Max Table Height` setting, the two buttons that have been created below the table will be positioned—in the output—directly below the 50% table (*see screenshot below*). We have also set the table's `Scroll Vertically` property to *Rows except header and footer*. This ensures that the header and footer are kept fixed in the fixed-height table, while the body rows scroll (*see screenshot below*).

**Note:** In the MobileTogether Designer Simulator, use the scroll wheel to scroll vertically, and click-and-drag to scroll horizontally.



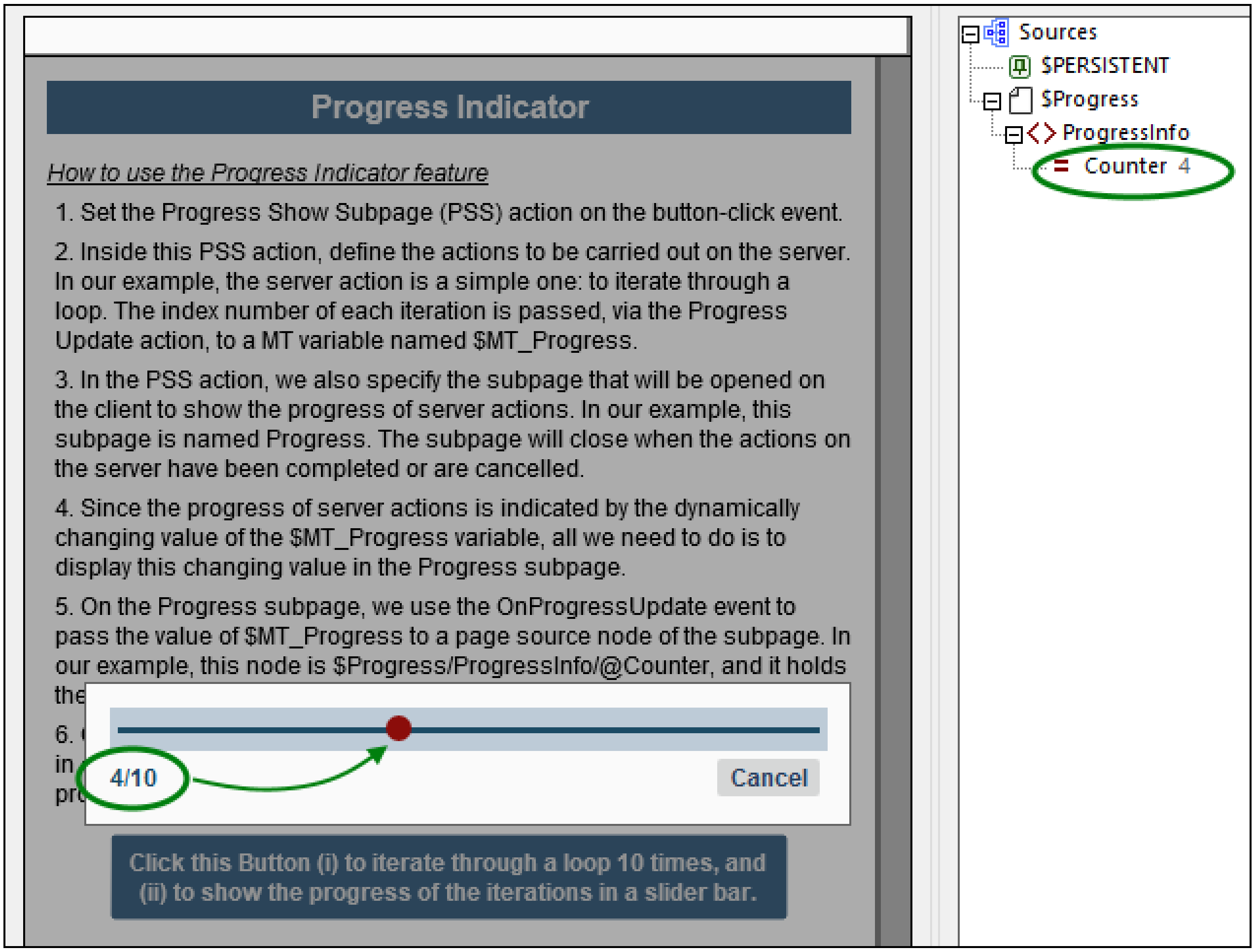
| Name | Order |
|---|---|
| Customer-01 | Some Item |
| Customer-02 | Some Item |
| Customer-03 | Some Item |
| Customer-04 | Some Item |
| Customer-05 | Some Item |
| Customer-06 | Some Item |
| Customer-07 | Some Item |
| Customer-08 | Some Item |
| Customer-09 | Some Item |
| Customer-10 | Some Item |
| Customer-11 | Some Item |
| Customer-12 | Some Item |
| Customer-13 | Some Item |
| Customer-14 | Some Item |

Total number of customers: 29

You can modify the values of properties to test the various possibilities. See the section Table Properties[1085] for details of scrollable table properties.

## 4.11 Progress Indicator

The Progress Indicator tutorial shows you how to create a progress indicator on client devices that will display how far a set of actions on the server has progressed. The tutorial will explain how the different components of the Progress Indicator feature should be used together. Broadly, the progress indicator is shown on a progress subpage that appears when the server actions start to be executed. The progress subpage closes automatically when execution of the server actions is completed—or when the client user cancels execution.



Progress Indicator

*How to use the Progress Indicator feature*

1. Set the Progress Show Subpage (PSS) action on the button-click event.

2. Inside this PSS action, define the actions to be carried out on the server. In our example, the server action is a simple one: to iterate through a loop. The index number of each iteration is passed, via the Progress Update action, to a MT variable named $MT_Progress.

3. In the PSS action, we also specify the subpage that will be opened on the client to show the progress of server actions. In our example, this subpage is named Progress. The subpage will close when the actions on the server have been completed or are cancelled.

4. Since the progress of server actions is indicated by the dynamically changing value of the $MT_Progress variable, all we need to do is to display this changing value in the Progress subpage.

5. On the Progress subpage, we use the OnProgressUpdate event to pass the value of $MT_Progress to a page source node of the subpage. In our example, this node is $Progress/ProgressInfo/@Counter, and it holds the

6. (

in

pr

4/10

Cancel

Click this Button (i) to iterate through a loop 10 times, and (ii) to show the progress of the iterations in a slider bar.

The components of the Progress Indicator feature are the following:

- A Progress Show Subpage action[795], which (i) specifies the subpage that will be displayed on the client to indicate the progress of server actions; and (ii) defines, as its child actions, the server actions to carry out for which a progress indicator is required.
- A Progress Update[796] action, which specifies what value to pass to the dynamically responsive **`$MT_Progress`**[1304] variable.
- The dynamic **`$MT_Progress`**[1304] variable, which, through its changing value, is used to indicate the progress of server actions.
- A page event named **`OnProgressUpdate`**[401], which is triggered by the Progress Update[796] action and can be used to update a progress subpage with information about server-action progress (via the `$MT_Progress` variable).

- A Progress Send Cancellation[797] action, which, when triggered, sets the value of the **`mt-progress-cancellation()`**[1262] function to `true()`.
- The **`mt-progress-cancellation()`**[1262] function, which can be used to test whether the client has sent a cancellation request or not.

## The tutorial file

The design file that you will get after completing this tutorial should look something like the following file located in your *(*[72] *My) Documents*[72] MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\ProgressIndicator`**.

`ProgressIndicator.mtd`: This is the completed MobileTogether design file. To see what it does, run a simulation in MobileTogether Designer by pressing **F5**.

You can start from scratch and create the file by following the steps of the tutorial. Alternatively, you can open the completed design file and refer to it while reading this tutorial.

## Tutorial structure

This tutorial is organized into the following sections:

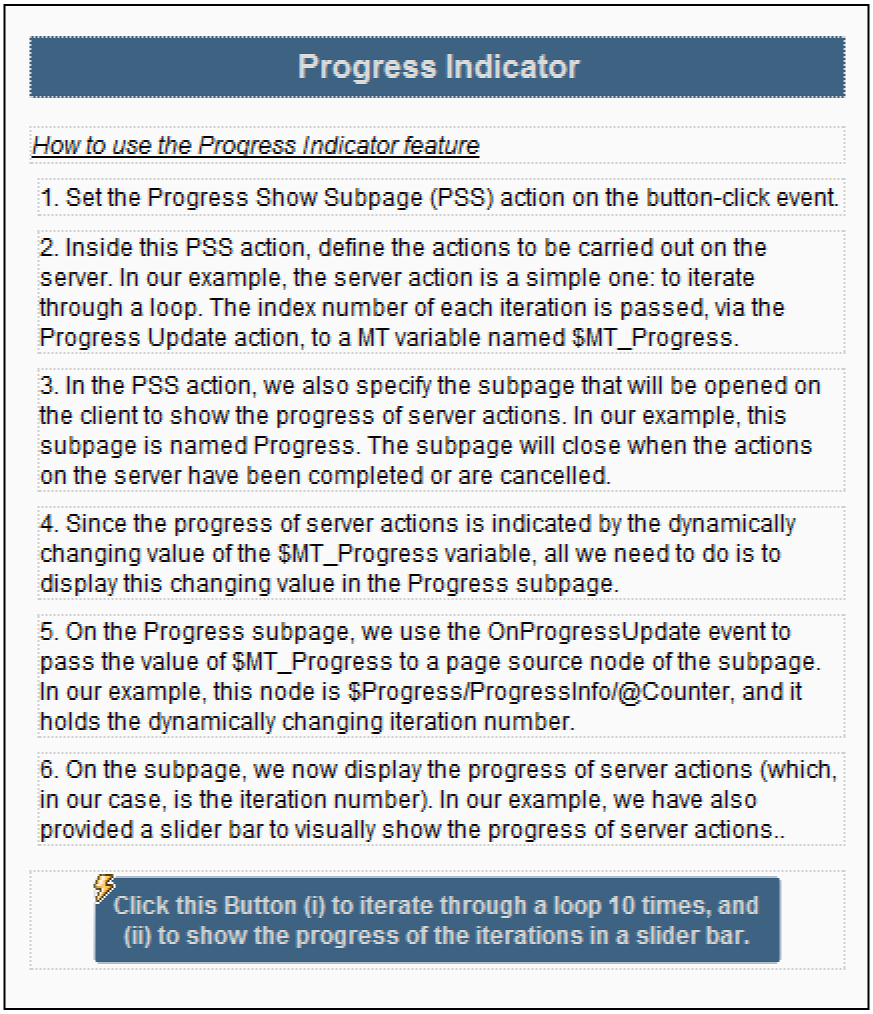
- Main Page[242] describes the main page settings. Essentially these are: (i) definitions of the server actions for which a progress indication is required, and (ii) the call to the subpage that will show the progress report.
- Progress Subpage[247] shows how to set the components that are needed to show the progress indicator on the progress subpage.

# 4.11.1 Main Page

The main page is what the solution opens with. On it we will place a button that (i) starts a set of server actions and (ii) opens a subpage that displays the progress of the server actions.

In order to keep things simple, we restrict the server actions to only iterate over a loop. Within each iteration, we do nothing except send a progress update.

In our example[241] (*screenshot below*), we have added labels that provide the header of the page and that list the broad steps of the Progress Indicator mechanism. You do not need these labels, but you do need the button that is at the bottom of the page.



Progress Indicator

How to use the Progress Indicator feature

1. Set the Progress Show Subpage (PSS) action on the button-click event.

2. Inside this PSS action, define the actions to be carried out on the server. In our example, the server action is a simple one: to iterate through a loop. The index number of each iteration is passed, via the Progress Update action, to a MT variable named $MT_Progress.

3. In the PSS action, we also specify the subpage that will be opened on the client to show the progress of server actions. In our example, this subpage is named Progress. The subpage will close when the actions on the server have been completed or are cancelled.

4. Since the progress of server actions is indicated by the dynamically changing value of the $MT_Progress variable, all we need to do is to display this changing value in the Progress subpage.

5. On the Progress subpage, we use the OnProgressUpdate event to pass the value of $MT_Progress to a page source node of the subpage. In our example, this node is $Progress/ProgressInfo/@Counter, and it holds the dynamically changing iteration number.

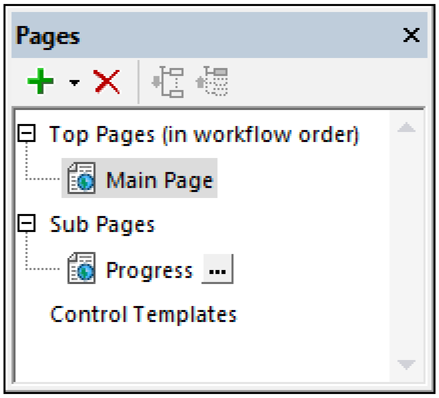
6. On the subpage, we now display the progress of server actions (which, in our case, is the iteration number). In our example, we have also provided a slider bar to visually show the progress of server actions..

Click this Button (i) to iterate through a loop 10 times, and (ii) to show the progress of the iterations in a slider bar.

This topic explains how to create the main page of your solution. The next topic, Progress Subpage 247, describes how to create the subpage that contains the progress indicator.

## Main page and subpage

A top page is created by default when you start a new design, and it is listed in the Pages Pane 257. Rename your top page by double-clicking it and typing a suitable name. In our example, we have renamed the top page to *Main Page* (*see screenshot below*).



To add a subpage, click the dropdown arrow of the *Plus* icon in the toolbar and select **Add Sub Page**. Name the subpage *Progress*.

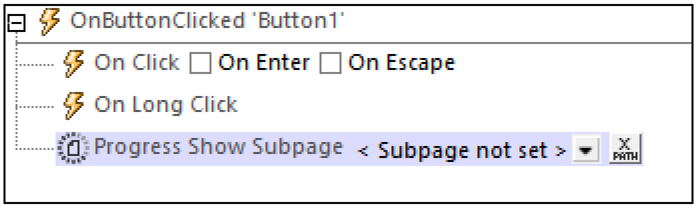
## Add the Progress Show Subpage action

The Progress Show Subpage[795] action does the following:

- Specifies the subpage to use for the display of server-action progress
- Takes, as its child actions, the server actions for which the progress indicator is required

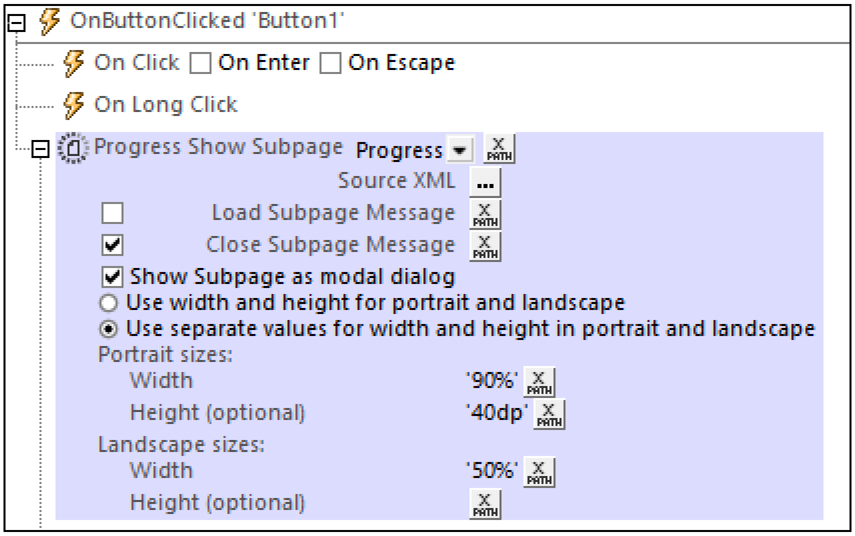
We will set up the Progress Show Subpage[795] action on a button-click event. Do this as follows:

1. Add a button by dragging a button control from the Controls Pane[266]. Double-click the button and enter a button name or other suitable text.
2. Right-click the button and select **Control Actions for OnButtonClicked**.
3. Drag the Progress Show Subpage[795] action into the main pane (*see screenshot below*) from the set of Page actions in the left pane.


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Progress Show Subpage < Subpage not set >

## Subpage settings

In the Progress Show Subpage[795] action, click the dropdown arrow and select the *Progress* subpage. Since we want to display the subpage as a modal dialog (one which overlays the main page), select the *Modal dialog* option and define the modal dialog's dimensions as you like (*see screenshot below*).



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

Progress Show Subpage Progress

Source XML ...

□ Load Subpage Message

☑ Close Subpage Message

☑ Show Subpage as modal dialog

○ Use width and height for portrait and landscape

● Use separate values for width and height in portrait and landscape

Portrait sizes:

Width '90%'

Height (optional) '40dp'

Landscape sizes:

Width '50%'

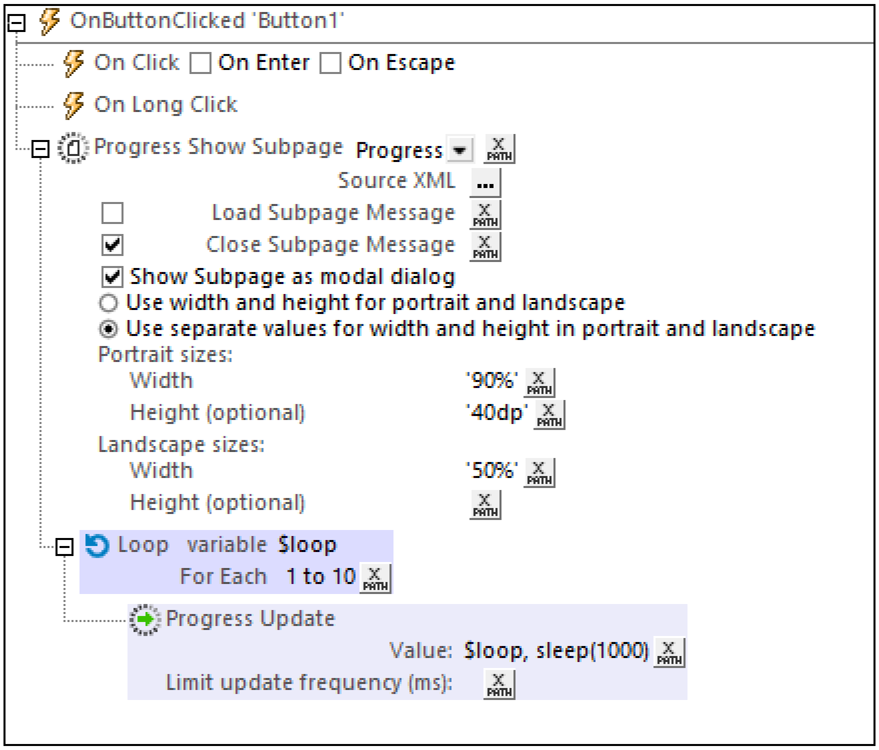
Height (optional)

With these settings, you have specified that, when the button is clicked, the *Progress* subpage will be opened as a modal dialog. The next step will be to specify the server actions for which you want progress reported.

## Server actions and the Progress Update action

The server actions for which a progress indicator is required must be defined as child actions of the Progress Show Subpage[795] action.

Since our server action will be a loop that iterates from 1 to 10, drag and drop the Loop[897] action into the main pane as a child of the Progress Show Subpage[795] action (*see screenshot below*). Give the action's variable a value of **`$loop`**. This will mean that for each iteration of the loop, the value of the **`$loop`** variable will be the index number of the current iteration (for example, **4** during the fourth iteration).



OnButtonClicked 'Button1'

On Click ☐ On Enter ☐ On Escape

On Long Click

Progress Show Subpage Progress

Source XML ...

☐ Load Subpage Message

☑ Close Subpage Message

☑ Show Subpage as modal dialog

○ Use width and height for portrait and landscape

● Use separate values for width and height in portrait and landscape

Portrait sizes:

| | |
|---|---|
| Width | '90%' |
| Height (optional) | '40dp' |

Landscape sizes:

| | |
|---|---|
| Width | '50%' |
| Height (optional) | |

Loop variable $loop

For Each 1 to 10

Progress Update

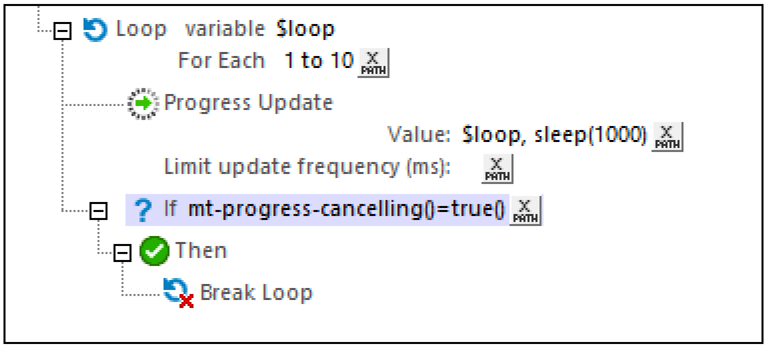
Value: $loop, sleep(1000)

Limit update frequency (ms):

Now add a Progress Update[796] action as a child of the Loop[897] action. This will cause the Progress Update[796] action to be executed once within each iteration. Importantly, the Progress Update[796] action sends a value, via its *Value* setting, to the **`$MT_Progress`**[1304] dynamic variable. In our example, we enter the XPath expression `$loop, sleep(1000)` (shown in the screenshot above) as the *Value* setting. This expression causes, during each iteration, the index number of the current iteration to be passed to **`$MT_Progress`**[1304]. Consequently, as the iterations proceed, the value in **`$MT_Progress`**[1304] will change from 1 to 10 till all the iterations have been completed. (The XPath function **`sleep()`** pauses the iteration for 1 second, which will slow the progress indicator on the subpage sufficiently enough for us to observe the progress.) When all the iterations have been completed, the loop is exited and the Progress Show Subpage[795] action is ended—which has the effect of closing the subpage.

## Cancel server actions

To allow for the user wanting to cancel the server actions, we build an option to exit the loop if the Progress Send Cancellation[797] action has been executed on the subpage[247]. When a cancellation action is executed, the globally available MobileTogether extension function **`mt-progress-cancelling()`**[1262] is set to **`true()`** from its default of **`false()`**. We can therefore test the value of this function and, if it is true, we can set up a cancellation procedure. In our example, the cancellation procedure is to break the loop as shown in the screenshot below.



```
Loop   variable $loop
       For Each   1 to 10
   Progress Update
                Value:  $loop, sleep(1000)
       Limit update frequency (ms):
   If  mt-progress-cancelling()=true()
      Then
         Break Loop
```

In the next section, Progress Subpage[247], we will use the value sent by the Progress Update[796] action to **$MT_Progress**[1304] to create the display of the progress indicator.

## 4.11.2 Progress Subpage

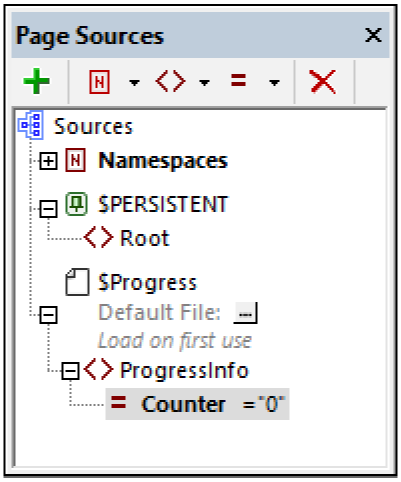
In our example, the subpage to display the progress of server actions is called *Progress*. The server actions are the iterations of a loop. To indicate the progress of the server actions, we will show the index number of the current iteration as the iterations progress. Additionally, we will show the progress graphically in a Horizontal Slider[520] control.

We will need to set up the following:

- A page source node to hold the iteration number as it changes. For this, we will use the OnProgressUpdate[401] subpage event and the **$MT_Progress**[1304] dynamic global variable
- The progress indicators (a number display and a slider control)
- Additionally, we will set up a Cancel button that will enable the user to cancel the server actions; the Cancel mechanism uses the Progress Send Cancellation[797] action and the **mt-progress-cancelling()**[1262] function.

### Progress information via OnProgressUpdate and $MT_Progress

First, in the Page Sources Pane[270], create an XML page source named **$Progress** as in the screenshot below. Name the root element **ProgressInfo**, and give it an attribute named **Counter**. Right-click **@Counter** and give it a fixed value of **0**. Next, right-click the page, select **Page Actions**, and, for the OnPageLoad[390] event, add an Update Node(s)[886] action that updates the **@Counter** node with a value of **0**. With these steps, we have created the **@Counter** node and ensured that it will have a value of **0** each time the subpage is loaded.

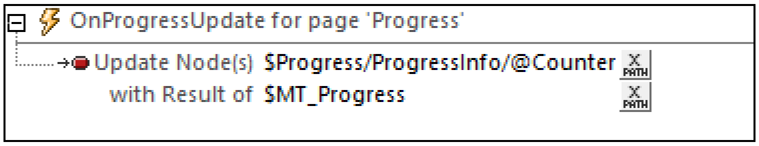


Page Sources

Sources
Namespaces
$PERSISTENT
Root
$Progress
Default File:
Load on first use
ProgressInfo
Counter ="0"

Now we need to pass the iteration number, which is stored in the **`$MT_Progress`**[1304] variable (*see previous topic*), to the **`@Counter`** node. We do this as follows:

1. Right-click the page and select **Page Actions**.
2. Select the the OnProgressUpdate[401] event.
3. Add an Update Node(s)[886] action by dragging the control from the left pane into the main pane.
4. Set the node that will be updated to be **`@Counter`** .
5. Set the update value to **`$MT_Progress`**[1304] (*see screenshot below*).



OnProgressUpdate for page 'Progress'
Update Node(s) $Progress/ProgressInfo/@Counter
with Result of $MT_Progress

These steps ensure that the **`@Counter`** node is updated with each iteration to contain the number of the current iteration.

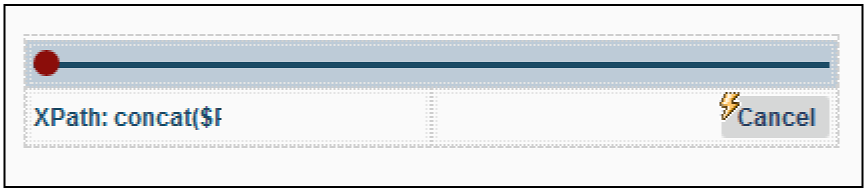
The background mechanism works as follows:

1. Each time the Progress Update[796] action inside an iteration is executed, (i) the current iteration number is passed to the **`$MT_Progress`**[1304] variable, and (ii) the subpage event OnProgressUpdate[401] is triggered. *See previous topic*[242].
2. Each time the OnProgressUpdate[401] event is triggered, the **`@Counter`** node is updated with the value in the **`$MT_Progress`**[1304] variable, which is the number of the current iteration, *See above*.

## Progress indicators

We will have two progress indicators: (i) a horizontal slider, and (ii) a number indicator.

In our design, we have created a table with two rows (*screenshot below*). Create a table with the same structure: two rows and two columns, with the top row spanned across both columns.



*Horizontal slider*
Do the following:

1. Drag and drop a Horizontal Slider[520] control into the top row.
2. In the Styles & Properties Pane[274], set the slider's *Min Value* to 0, and its *Max Value* to 10. Optionally, set the slider's colors as you like.
3. Drag and drop the **`@Counter`** node onto the slider. This associates the slider with the **`@Counter`** node (which contains the dynamically changing iteration number).

At run time, as the value in the **`@Counter`** node changes, the slider will, because of its association with the **`@Counter`** node, move to the new value on the slider scale.

*Number indicator*
Do the following:

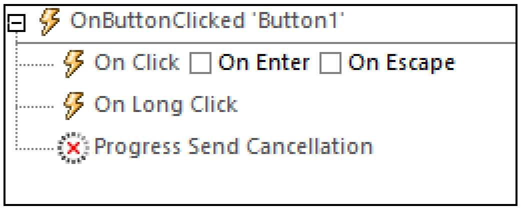
1. Drag and drop a Label[554] control into the left-hand cell of the second row (*see screenshot above*).
2. In the Styles & Properties Pane[274], set the label's *Text* property to the XPath expression `concat($Progress/ProgressInfo/@Counter, '/10')`. Optionally, set colors and text size as you like.

At run time, as the value in the **`@Counter`** node changes, the value will be displayed in the label in the form `X/10`.

## Canceling server actions

To enable the user to cancel server actions, we add a **Cancel** button to the right-hand cell of the second row (by dragging a Button[417] control there, double-clicking the button and typing *Cancel*).

The Cancel action is set by adding the Progress Send Cancellation[797] action to the button's OnButtonClicked event (*see screenshot below*). (You can access this event via the button's context menu).


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Progress Send Cancellation

When the button is clicked, the Progress Send Cancellation[797] action is executed, which causes the **`mt-progress-cancelling()`**[1262] function to be set to **`true()`**. In the definition of the server actions (*see previous topic*[242]), the value of this function can be used as a test. If the value is **`true()`**, then a cancellation procedure can be initiated. In our example, the cancellation procedure consists of breaking the iteration loop.

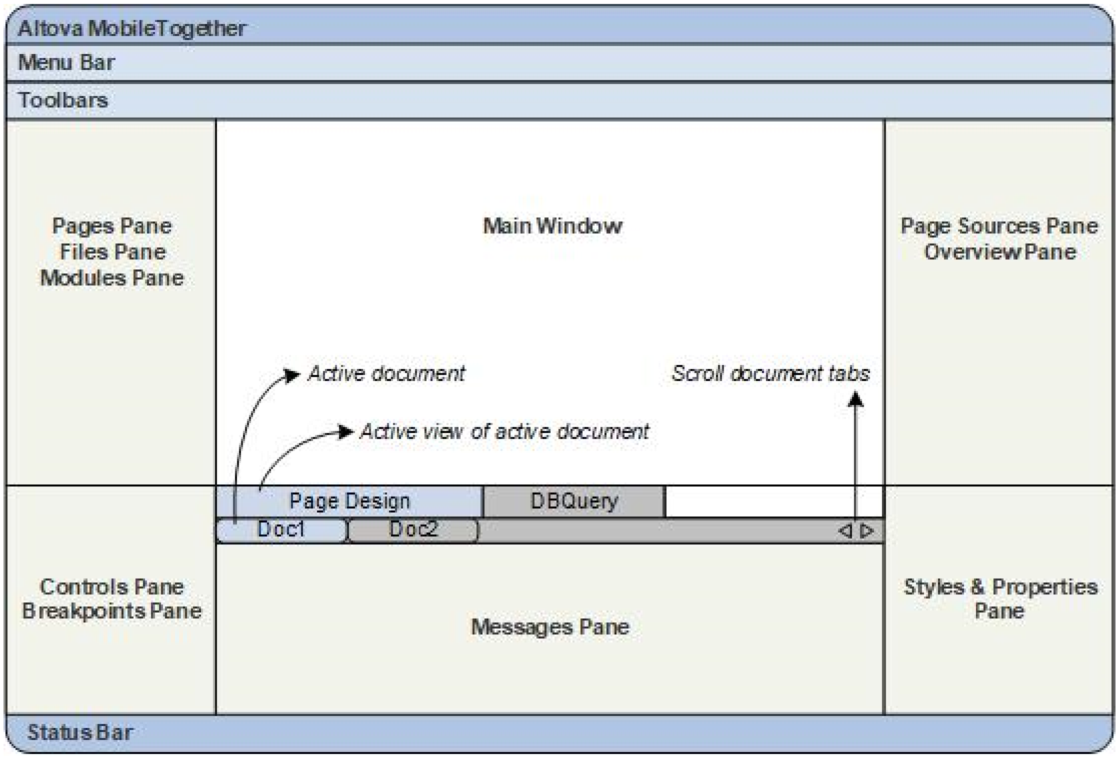
**Note:** You can use the `mt-progress-cancelling()` 1262 function not only to run a cancellation procedure on the server, but also to run a cancellation procedure on the client (that is, on the subpage). For example, you might want to display a cancellation message for the user while the cancellation procedure is running on the server.

## Simulation

After you have completed your design, you can test it by running a simulation: press **F5**. You can subsequently use the design to try out other progress indication scenarios.

# 5 User Interface

The Graphical User Interface (GUI) consists of a Main Window[253] (consisting of the Page Design[253] and DB Query[255] View tabs) and several panes (*see screenshot below*). By default, the panes are located around the Main Window, but can be moved around the GUI, minimized, or hidden.



The panes, which are listed below, are described in the sub-sections of this section:

- Pages Pane[257]
- Files Pane[259]
- Controls Pane[266]
- Modules Pane[263]
- Breakpoints Pane[269]
- Page Sources Pane[270]
- Overview Pane[272]
- Styles & Properties Pane[274]
- Messages Pane[278]
- Listings Pane[281]
- Find & Replace Pane[283]

## Showing/hiding panes

A pane can be displayed or hidden by toggling it, respectively, on or off in the **View** menu. A displayed pane can also be hidden by right-clicking its title bar of the displayed pane and selecting the command **Hide**.

## Floating and docking the panes

An individual pane can either float free of the GUI or be docked within the GUI. When a floating window is docked, it docks into its last docked position. A window can also be docked as a tab within another window.

A window can be made to float or dock using one of the following methods:

- Right-click the title bar of a window and choose the required command (**Floating** or **Docking**).
- Double-click the title bar of the window. If docked, the window will now float. If floating, the window will now dock in the last position in which it was docked.
- Drag the window (using its title bar as a handle) out of its docked position so that it floats. Drag a floating window (by its title bar) to the location where it is to be docked. Two sets of blue arrows appear. The outer set of four arrows enables docking relative to the application window (along the top, right, bottom, or left edge of the GUI). The inner set of arrows enables docking relative to the window over which the cursor is currently placed. Dropping a dragged window on the button in the center of the inner set of arrows (or on the title bar of a window) docks the dragged window as a tabbed window within the window in which it is dropped.

To float a tabbed window, double-click its tab. To drag a tabbed window out of a group of tabbed windows, drag its tab.

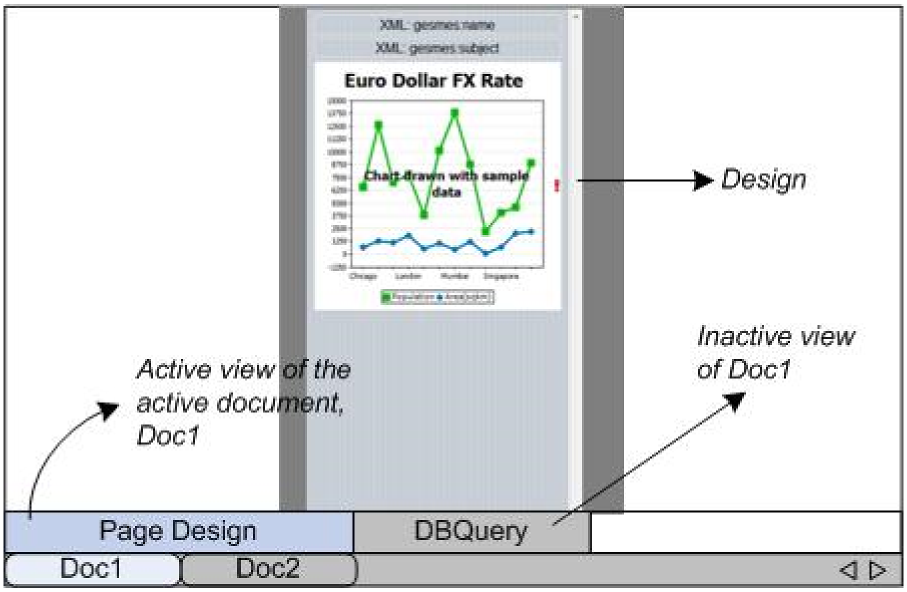
## Auto-hiding panes

The Auto-hide feature enables you to minimize docked panes to buttons along the edges of the application window. This gives you more screen space for the Main Window and other panes. Scrolling over a minimized pane rolls out that pane.

To auto-hide and restore panes click the drawing pin icon in the title bar of the pane window (or right-click the title bar and select **Auto-Hide**).

# 5.1 Main Window

The Main Window (*screenshot below*) is where you design the pages of the MobileTogether Designer project and directly query a database to preview table data. It consists of two views, only one of which can be active at a time: Page Design 253 and DB Query 255.



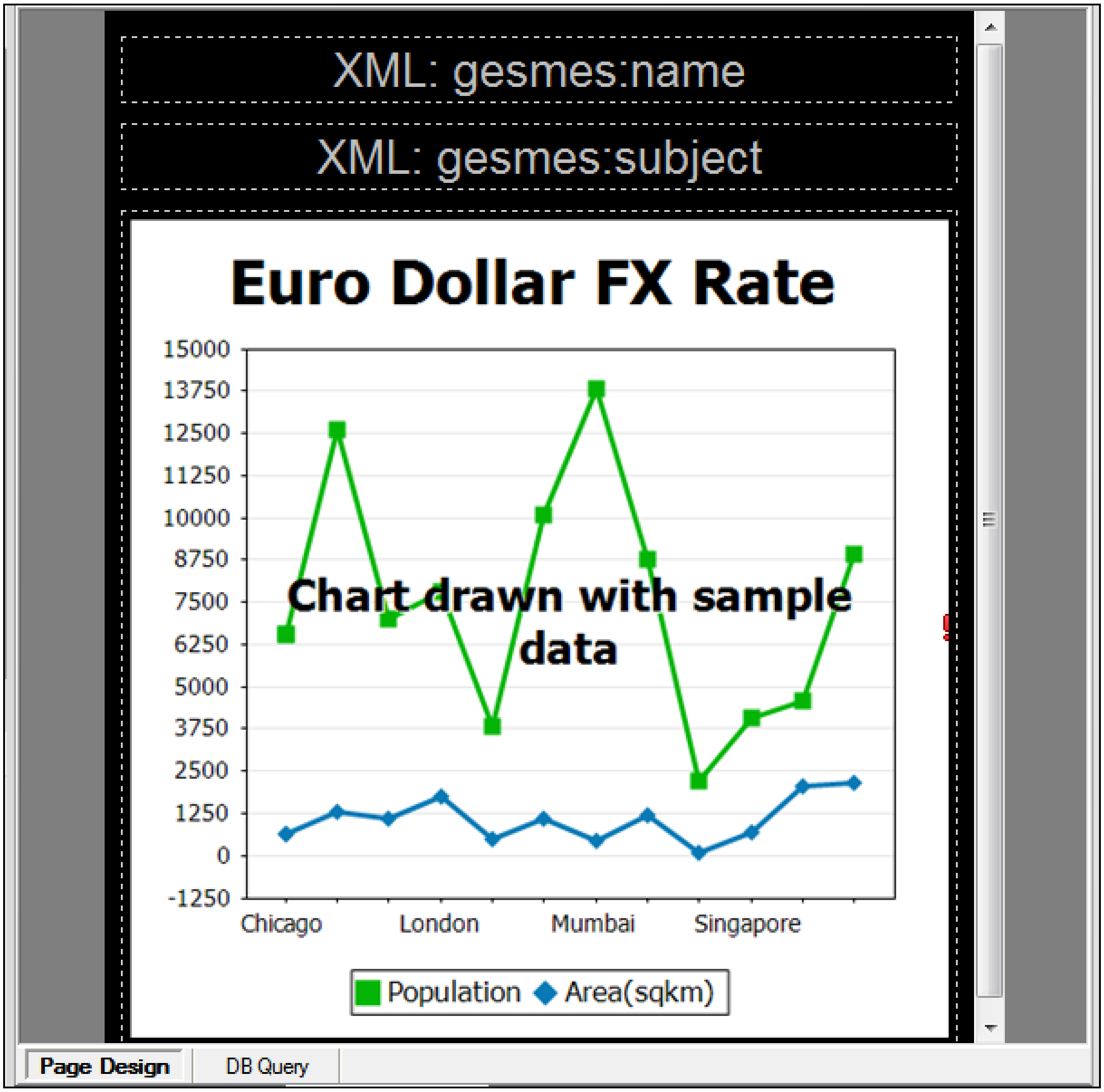
## MobileTogether Design (.mtd) files in the Main Window

Note the following points:

- Any number of MobileTogether Design (`*.mtd`) files can be open simultaneously. You can switch among the open documents and edit them.
- Each open document has its own window and a tab with its name at the bottom of the Main Window. To make an open document active, click its tab.
- If several files are open, some document tabs might not be visible for lack of space in the document tabs bar. Document tabs can be brought into view by: (i) using the scroll buttons at the right of the document tab bar, or (ii) selecting the required document from the list at the bottom of the Window menu.
- You can activate open files in the sequence in which they were opened by using **Ctrl+Tab** or **Ctrl+F6**.
- Right-clicking a file tab opens a context-menu that contains a selection of **File** 1562 commands, such as **Print** 1582 and **Close** 1567.
- Placing the mouse cursor over components in the Main Window displays a popup containing further information about the function of that component.

## 5.1.1 Page Design

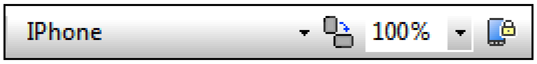
The **Page Design View** (**Page View** for short) is the view in which the page that is going to be deployed to the mobile device is designed (*see screenshot below*).



To design a page, drag-and-drop controls from the Controls Pane[266] into the design, and then specify, in the Properties Pane[274], the settings of that control. Controls can be positioned anywhere on the page. As you drag a control over the page, possible drop positions are indicated with an arrow. The settings of a control can be edited at any later time by selecting the control in the design and editing its settings in the Properties Pane[274]. Delete a control in the design by selecting it and pressing **Delete**.

## Page view and device settings

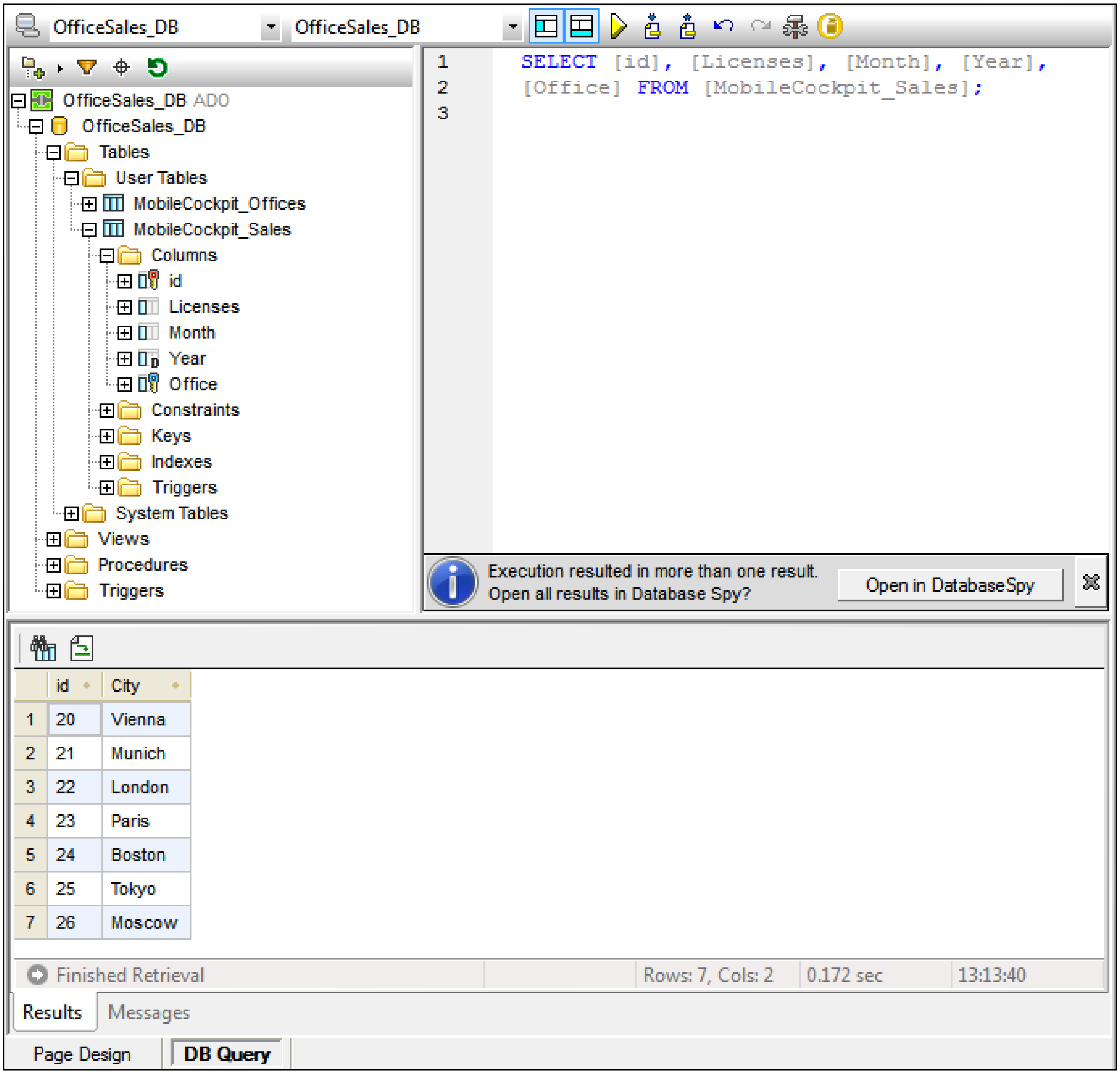
The screenshot below shows the Page View settings in the Main toolbar. The settings you make here also determine the settings of the device used in simulations[1355].



- *Preview Device:* This combo box allows you to select various mobile devices, so that the design can be previewed for specific devices. The selected device will be not only the device used in the design, but also that used in simulations. You can change the preview device at any time.
- *Portrait/Landscape Preview Toggle:* Toggles the design preview between portrait and landscape.
- *Zoom level:* A combo box to select the zoom level in 10% steps from 10% to 100%. The zoom level can also be modified via the **View**[1651] menu[1651].
- *Lock all page views to same device and zoom level:* The page views of all open documents will be locked to the currently selected device and zoom level.

## 5.1.2 DB Query

The **DB Query View** (*screenshot below*) enables you to directly query any major database from within the MobileTogether Designer GUI. The database could be a page source referenced in the active document or an external database. Note that each DB Query pane is associated with the currently active design. You can have connections to multiple databases for a single design. There can also be multiple designs open in MobileTogether Designer. Queries and actions defined in the DB Query View are independent of other MobileTogether Designer tabs, and are not saved as part of the `.mtd` design file.



```
SELECT [id], [Licenses], [Month], [Year],
[Office] FROM [MobileCockpit_Sales];
```

Execution resulted in more than one result.
Open all results in Database Spy?

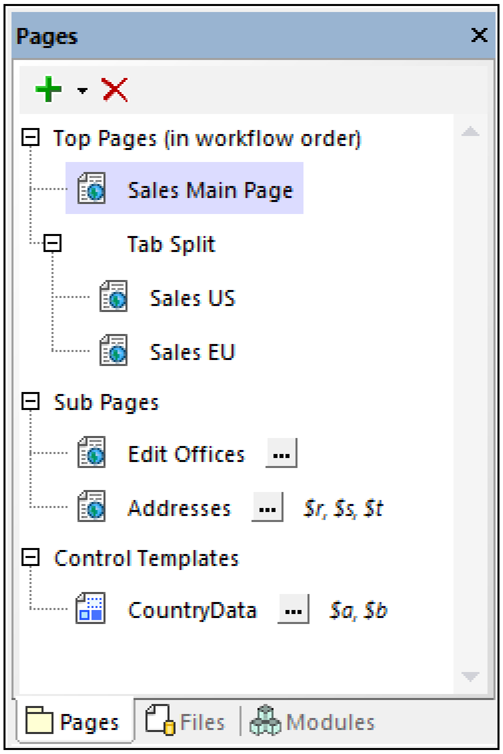
| | id | City |
|---|---|---|
| 1 | 20 | Vienna |
| 2 | 21 | Munich |
| 3 | 22 | London |
| 4 | 23 | Paris |
| 5 | 24 | Boston |
| 6 | 25 | Tokyo |
| 7 | 26 | Moscow |

Finished Retrieval | Rows: 7, Cols: 2 | 0.172 sec | 13:13:40

See the section, Database Query[1046], for a detailed description.

## 5.2 Pages Pane

The **Pages Pane** (*see* *User Interface*[251] *for its location*) enables you to add new pages to a project and displays a tree of all the pages in the project (*see screenshot below*). To see the default location of the Pages Pane, go to The Graphical User Interface (GUI)[251].



To display a page, click its entry in the Pages Pane.

### Top pages, sub pages, tabbed pages, and control templates

There are four kinds of page-type components at the project level:

- *Top pages:* A top page is part of the sequence of pages that makes up a workflow. When the solution is started, it progresses from the first page to the last, in the order listed in the Pages Pane. You can change the position of a page in the list by dragging the page to a new position. Tabbed pages are also part of the workflow sequence, but sub pages are not.
- *Tabbed pages (or tab splits):* A tabbed page (or tab split) is a page with tabs, each of which, contains a page. For example, in the screenshot above, the tabbed page (indicated by *Tab Split*) is defined to have two tabs that contain, respectively, the pages *Sales US* and *Sales EU*. Tabbed pages are part of the page sequence that makes up the project's workflow.
- *Sub pages:* A sub page is not part of the sequence of pages that make up the workflow of the project. It is similar to a module that is called by a control in a top page (or tabbed page). For example, an `OnButtonClicked` event of a Button control in a top page could use the `GoToSubpage` action to go to a particular subpage, and then return to the top page (or go to another page). You can declare parameters and variables for a sub page by clicking its **Add Parameters/Variables** button.
- *Control templates:* A control template resembles a page in that you can lay out a set of controls on it. It is called a template because it can be reused (with all its controls) on other pages. You can declare

parameters and variables for a control template by clicking its **Add Parameters/Variables** button. See Control Templates (CTs)[1200] for more information.

Also see Pages, Tabbed Pages, and Sub Pages[381] for more information about these kinds of pages..

## Adding, renaming, and deleting pages

Besides the methods listed in the following table, you can also use context menu commands for some of these tasks (*see further below*).

| To... | Do this... |
|---|---|
| Add a page | Click the **Add Page** icon in the pane's toolbar. From the dropdown menu, select **Add Top Page**, **Add Tab Split**, or **Add Sub Page**. A new page is added to the Pages Pane with a name of *New Page X* or *Process Tab Split*, and the empty new page is displayed in the Main Window. |
| Rename a page | Double-click the name in the Pages Pane and edit the name. |
| Delete a page | Select the page you want to delete and click the **Delete** icon in the pane's toolbar, or press **Delete**. |

## Context menu

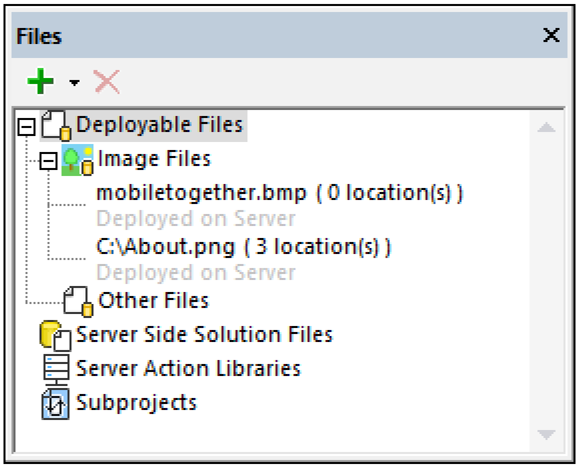
The context menu of items in the Pages Pane enables you to insert pages before the selected item (**Insert**), or to append pages (**Add**) to the sequence of top pages and to the list of sub pages. Child pages can only be added to tabbed pages via the context menu of the tabbed page item.

| To... | Do this... |
|---|---|
| Add a page before a top page, sub page, tabbed page, or child page of a tabbed page | Right-click the page and select **Insert Page** |
| Add a top page at the end of the *Top Pages* list | Right-click any item and select **Add Top Page** |
| Add a sub page at the end of the *Sub Pages* list | Right-click any item and select **Add Sub Page** |
| Add a child page at the end of a list of child pages of a tabbed page | Right-click the tabbed page and select **Add Page as Child** |
| Add a tabbed page before the selected page | Right-click and select **Insert Tab Split** |
| Add a tabbed page after the last page of the *Top Pages* list | Right-click in either list and select **Add Tab Split** |
| Add a control template before the selected page | Right-click and select **Insert Control Template** |
| Add a control template after the last control template | Right-click in either list and select **Add Control Template** |
| Find references in the design to the selected page | Right-click the page and select **List Usages** |

## 5.3 Files Pane

The **Files Pane** (*shown below; see User Interface*[251] *for its location*) enables you to do the following:

- Add files to the project that you want to deploy to the server and/or client when the project is deployed. Some files, such as the default files of page sources, are added automatically to the list of deployable files. But you can also add files directly via the context menu (obtained by right-clicking inside the Files Pane). The deployable files that you add here are read-only and can be image files and other files, such as XML files. They are organized in the pane under two headings: *Image Files* and *Other Files* (*see screenshot below*).
- Add server action libraries[1551] to the current project. A server action library contains an Action Group that provides a set of actions. Once a server action library has been added to a solution, its Action Groups become available for events of the solution.
- Add subprojects[1349] to the current project. After a subproject has been added to a project, its components become available for use in the project.



The context menu of the Files Pane contains some common commands for all three types of project-related files (deployable files, server action libraries, and subprojects), which are noted in the following list. Features that are specific to a given file type are described in the respective sections below.

- In the menu bar: the **Add** icon enables you to add a file, server action library, or subproject; the **Remove** icon enables you to remove the selected file, server action library, or subproject.
- For the context menu of each type of file, right-click the respective item in the Files Pane (*Deployable Files, Server Action Libraries,* or *Subprojects,* respectively).
- Make a filepath relative or absolute via the respective commands in the file's context menu.
- Copy the path of a selected file, server action library, or subproject. You can copy the path as listed in the window (after making the path relative or absolute), or you can copy the full absolute path.

*Adding and removing files, server action libraries, and subprojects*

| To... | Do this... |
|---|---|
| Add a file, server action library, or subproject | Right-click the respective item header in the Files Pane (*Deployable Files, Server Action Libraries*, or *Subprojects*) and, in the context menu that appears, click the respective **Add** or **Include** button, and then browse for the |

| | |
|---|---|
| | file you want[1563]. The added object will appear in the respective list. Alternatively, use the **Add** button in the menu bar. |
| Remove a file, server action library, or subproject | Right-click the file, server action library, or subproject and click the **Delete** command. Alternatively, select the file and click the **Delete** button in the menu bar. |

## Deployable Files

You can add a deployable file to the project by right-clicking *Deployable Files* and selecting the **Add Deployable File** command. Alternatively, you can right-click *Image Files* or *Other Files* and add deployable files via their respective context menus. The Files Pane provides the following features related to deployable files.

- You can select multiple deployable files to add in one selection.
- By default, a file will be added with its relative path. You can subsequently change this to an absolute path in the file's context menu.
- When you add a deployable file, you will be prompted about where you want to deploy the file: server, client or both. If you choose no option, the file will be added but will not be deployed at deployment time.
- Where a file will be deployed (server and/or client) is displayed below the file name. The deploy-location can be changed via the corresponding context menu command for that file.
- The total number of times a deployable file is used across all pages of the project is displayed next to the file name.
- Select a deployable file's **Open File** context menu command to open the file in the default application for its file type.
- List all usages of the selected deployable file via the corresponding command in the pane's context menu. This shows where in the design the deployed file is invoked and helps you to quickly locate the usage/s.

### About deployed files

Deployed files are read-only files that are stored on the server. Deploying is ideal for default XML files, image files, and other data files that will be used to only read data. If the data file is to be written to, then do not deploy it but store it on the server at a location that the MTD file correctly references. (See the sections Location of Project Files[289] and Deploying the Project[291] for more information.) Deploying a file subsequently with another design does not affect previously deployments.

+ Deploy only if the file is used as a read-only file.
+ Deploy if, for some reason, the file's URL will not be accessible to clients.
+ Deploy if you want the file in the same unchanged state when accessed by clients.
+ Deploy if file loading needs to be faster.
+ Deploy to the client if the solution references the file frequently.
– Do not deploy if the files needs to be written to.
– Do not deploy if large file-sizes on the server is an issue.
– Do not deploy if the file changes continuously and solutions require the latest version.
– Do not deploy if the design is to be sent to others; in this case, consider embedding data files in the design.

*Other ways to add deployable files to the Files Pane*
Files can also be added to the Files Pane in the following ways:

- When a file is added as a page source or when an image file is added to a page design, a dialog is displayed that asks whether the file should be deployed to MobileTogether Server. If you click **Yes**, the file is added to the Files Pane with its check box selected. If you click **No**, the file is added to the Files Pane with its check box unselected. You can select/deselect a file's check box in the Files Pane (*see screenshot above*) at any later time.
- In the context menu of the root node of a page source 359 in the Page Sources Pane 270.

⊟ Also see

Deploy to MobileTogether Server 1574
Location of Project Files 289
Deploying the Project 291
Data Storage on Servers 312

## Server side solution files

The Files Pane enables you to add files that you want to deploy to the *Server Side Solution's Working Directory* of MobileTogether Server. Note that, in the case of DBs, only file-based DBs can be added as server side solution files. The advantage of adding server side solution files to the project is that, at the time of deployment, these files will be automatically deployed to the server side solution's working directory. As a result, you do not need to manually save these files to the server. They will be automatically deployed together with the solution when the menu command **File | Deploy to MobileTogether Server** 1574 is selected.

To add a server side solution file, select the pane's toolbar command **Add Server Side Solution File**. You can also access this command in the pane's context menu (obtained by right-clicking in the pane).

For more information, see Location of Project Files 289, Deploying the Project 291, and Deploy to MobileTogether Server 1574.

## Server action libraries

The Files Pane provides a central place in which you can manage the server action libraries of your project.

- To add a server action library, use the **Add Server Action Library** command in either (i) the context menu of the *Server Action Libraries* item in the Files Pane, or (ii) the **Refactor** 1614 menu.
- If a server action library has been modified, reload it by right-clicking it in the Files Pane and selecting the **Reload** command.
- Remove a server action library by selecting it in the Files Pane and selecting **Remove** either in the context menu or the menu bar.

For further information about server action libraries, see the section Server Action Libraries 1551.

## Subprojects

In the Files Pane, you can include or import a subproject 1350, remove a subproject, and open a subproject in a new editing tab.

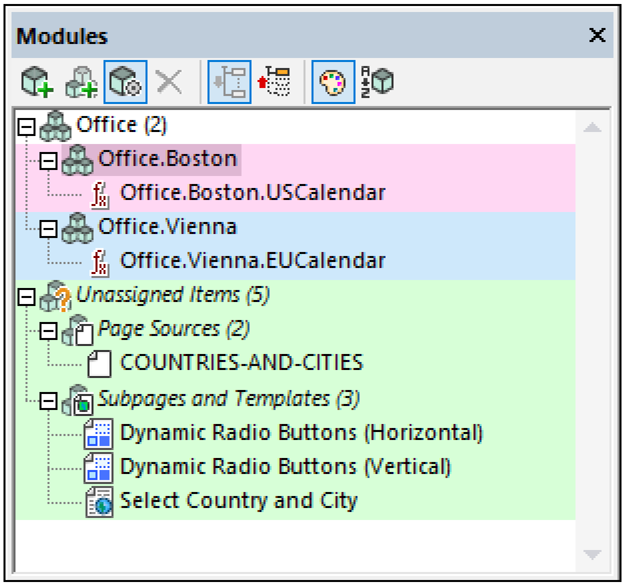
- To include a subproject, use the **Include Subproject** command in either (i) the context menu of the *Subprojects* item in the Files Pane, or (ii) the **Refactor** 1614 menu.

- Remove a server action library by selecting it in the Files Pane and selecting **Remove** either in the context menu or the menu bar.
- To quickly open a subproject for viewing or editing in a new tab, right-click the subproject in the Files Pane and select **Open Subproject Individually**.
- To import the components of an included subproject as components of the main project, right-click the included subproject in the Files Pane and select **Include Subproject As a Copy**. The components of the included subproject will be copied to the main project and the included subproject will no longer be referenced by the main project. The subproject's **`.mtd`** file, however, will not be deleted.

For further information about subprojects, see the section Subprojects 1349.

# 5.4 Modules Pane

The **Modules Pane** (*shown below; see* [*User Interface*](251) *for its location*) enables you to manage the modules of your project via (i) the context menu of the pane's items, (ii) the toolbar of the pane. This section describes these features of the pane. For a broader description, see the section [Modules](1352).



## The Unassigned Items module

The *Unassigned Items* module is present by default and contains all module items of the project that have not been assigned to a module (see [Modules](1352) for a list of module items). If a module item has a name that indicates to which module it belongs, then it will automatically be listed under the relevant in the Modules Pane. All other module items (those that do not have a [module name as part of its name](1352)) will be listed in the *Unassigned Items* module. To move an item from the *Unassigned Items* module to another module right-click the item, hover over **Modules**, and select the appropriate **Rename** command.

## Add and remove modules (and submodules)

These commands are available in the context menu and the toolbar. You can name or rename a module by double-clicking it or pressing **F2**.

- *Add a module*: Adds a top-level module.
- *Insert a module*: This command is available in the context menus of submodules (not top-level modules). It inserts a module at the same level as the selected submodule.
- *Add submodule*: Adds a submodule to the selected module.
- *Remove module*: Removes the selected module and all its contents.

**Note:** If you rename a module, then the names of all its items will be changed automatically to reflect the change.

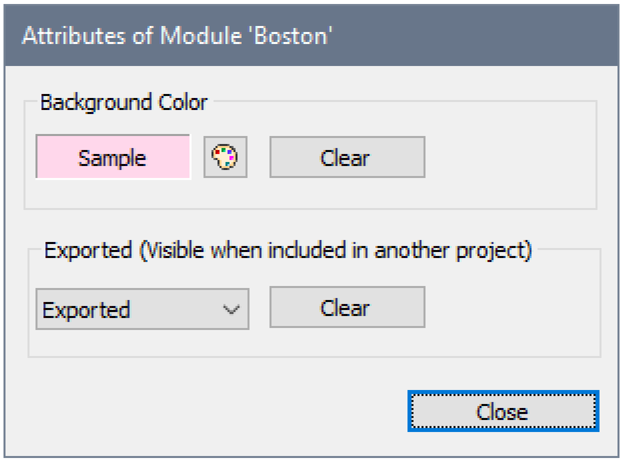
## Reassign items to other modules

You can reassign a module item to another module. Essentially it is the name of the item that determines to which module it belongs (see the section *Creating modules: module names and item names* 1352). So you can change the name of an item to assign it automatically to the appropriate module, or you can specify the target module via the mechanisms listed below (in which case the name of the item will be changed to effect the move).

- Right-click the item to move, hover over the Modules command of the context menu that appears, and select the relevant **Rename** command.
- Drag the module item you want to move to the new module and select the **Rename** command that appears.

## Module settings

You can assign a background color and an export setting to individual modules via the module's Attributes dialog (which is accessed via the **Module Settings** command of the module's context menu, *screenshot below*).



Attributes of Module 'Boston'

Background Color

Sample

Exported (Visible when included in another project)

Exported

Note the following points:

- The background color of a module is applied to all submodules unless a submodule has its own background color setting.
- When a subproject is extracted from a project 1350, all components of the project will, by default, be exported to the subproject. If you want to export only a subset of components, then you can group them in a module and set the *Export* property of the module (*see screenshot above*) to either *Exported* or *Not Exported* as appropriate. The *Not Set* value causes the value of the parent to be inherited, with the default value of top-level modules being *Exported*.

## Additional actions

Assign a color and an export setting to individual modules.

- Expand/collapse all modules via the respective toolbar buttons.
- To see the usages of a module item in the Listings Pane 281, right-click the item and select the command **List Usages**.

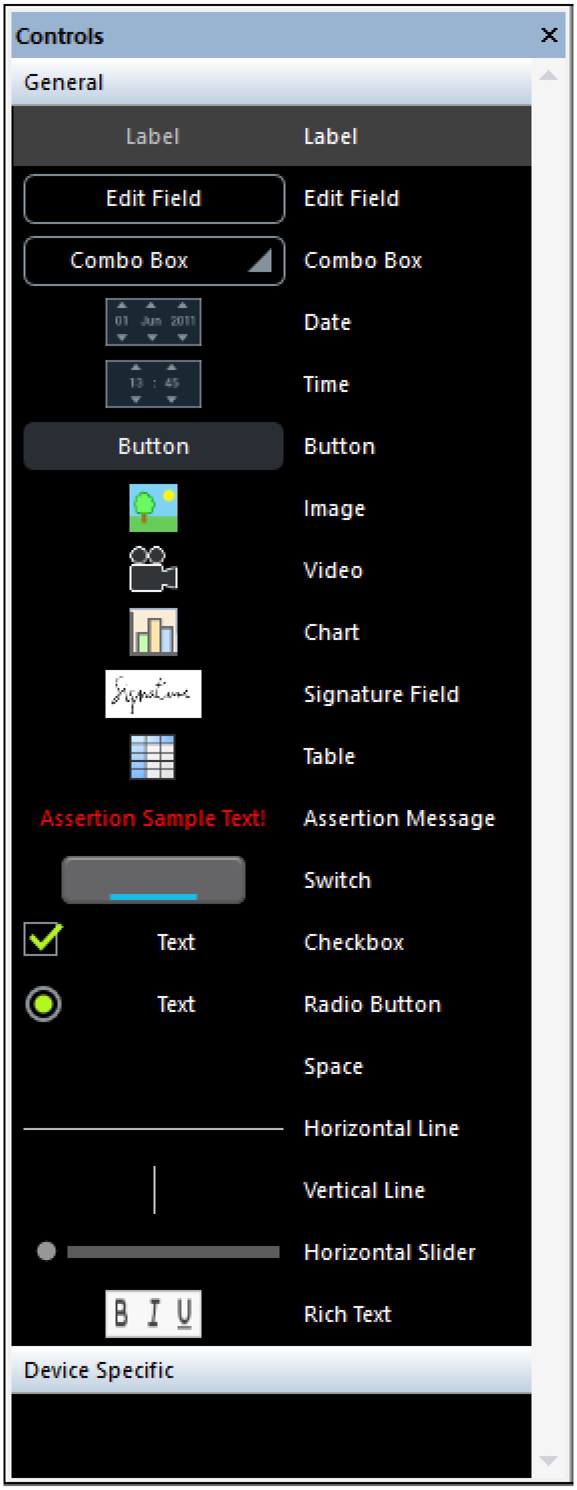
- Toggle on/off the colors, structure, and alphabetical sorting of modules via the respective toolbar buttons.

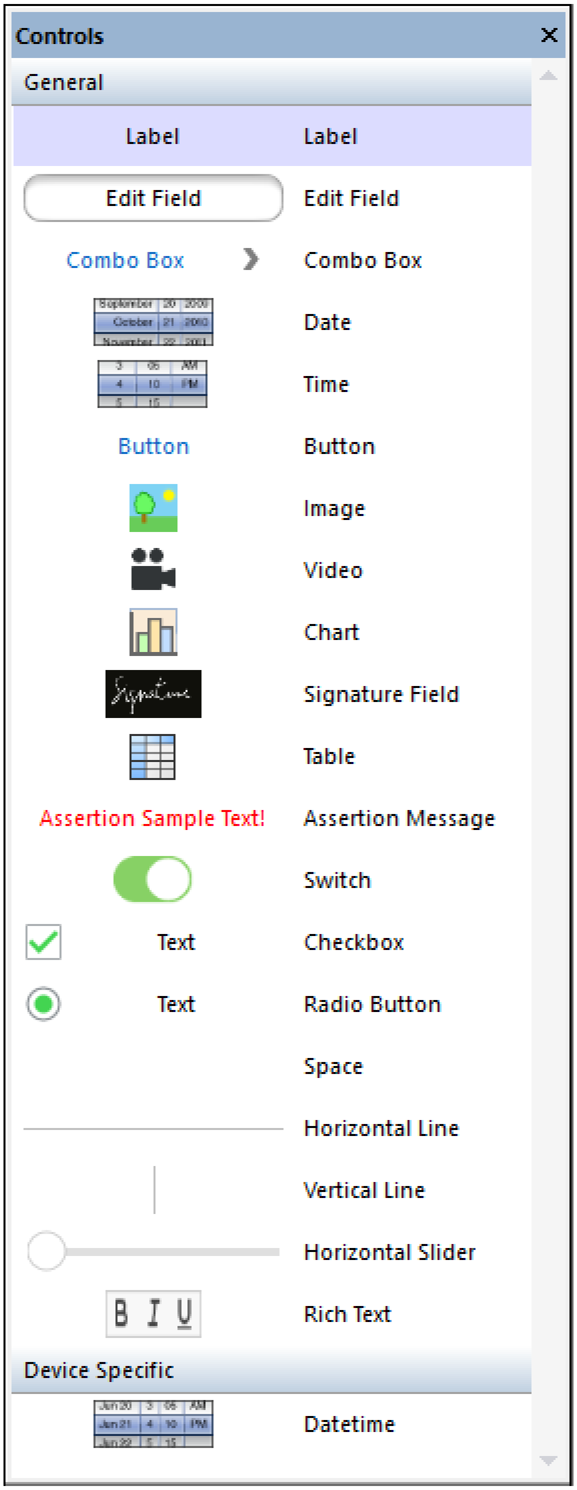
# 5.5 Controls Pane

The **Controls Pane** (*see User Interface* 251 *for its location*) shows all the controls that can be added to a page design or to a workflow. To see the default location of the Controls Pane, go to The Graphical User Interface (GUI) 251.

## Page design controls

Page design controls are available when the Page Design tab 253 is selected in the Main Window. The appearance of the Controls Pane and the controls corresponds to the currently selected device 253. For example, the Controls Panes shown in the screenshots below are for Android LG Optimus 7 (*left*) and iPhone 6 Plus (*right*).



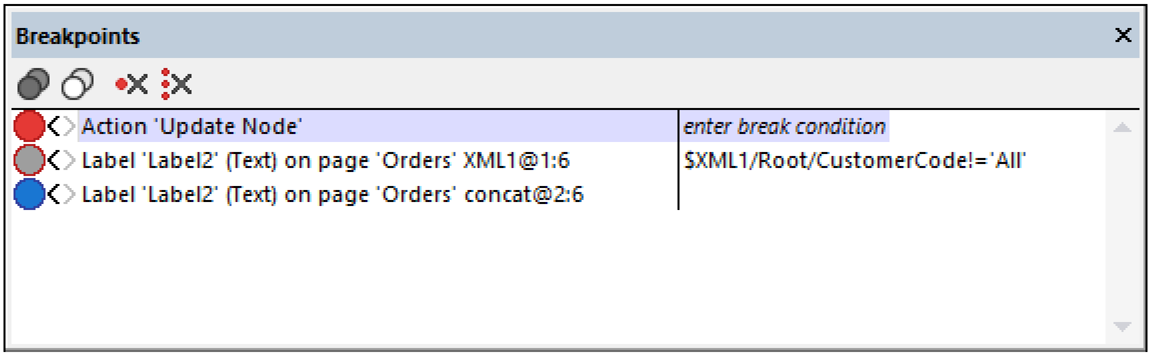


The pane's controls are organized into a *General* section and a *Device-Specific* section. To add a control to a page design, drag and drop the control onto the desired location in the page design[253].

In the context menu of a control (obtained by right-clicking the control), select the **List All** command to display in the Listings Pane[281] all instances of that control type that occur in the active design.

## 5.6 Breakpoints Pane

The **Breakpoints Pane** (*screenshot below; see User Interface*[251] *for its location*) lists all the project's breakpoints and tracepoints and provides a convenient single place to manage these debug points. Breakpoints can be set on actions[1390] and XPath expressions[1252], and are indicated by a solid red circle. Tracepoints can be set on XPath expressions[1252], and are indicated by a solid blue circle. Both types of debug points may be disabled, in which case they are ignored during debugging sessions; disabled debug points are indicated by a solid gray circle with a border color that indicates which type of debug point it is (red for breakpoints, blue for tracepoints).



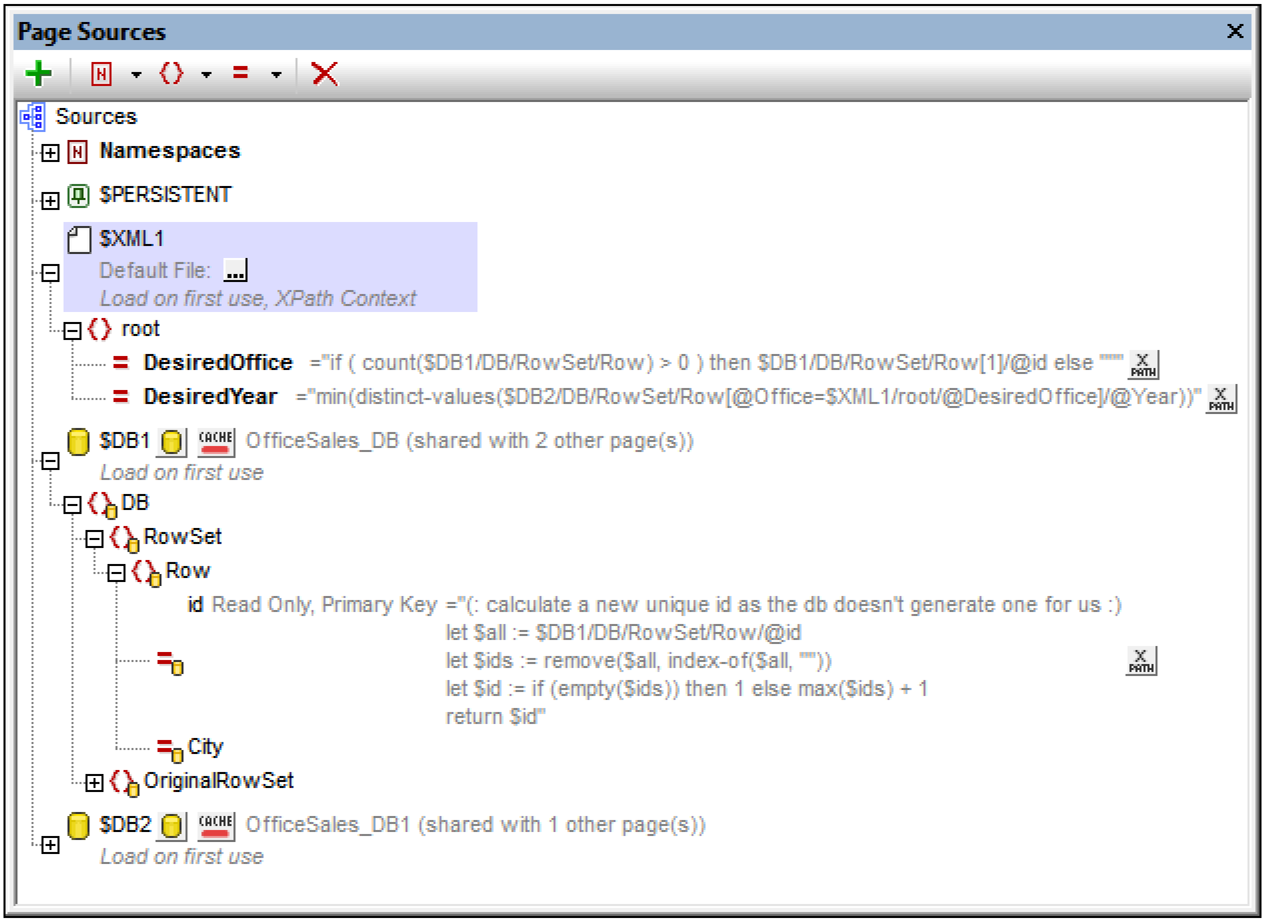
Action 'Update Node' *enter break condition*
Label 'Label2' (Text) on page 'Orders' XML1@1:6 $XML1/Root/CustomerCode!='All'
Label 'Label2' (Text) on page 'Orders' concat@2:6

The following features are available:

- To filter what types of debug points are shown, right-click in any free area of the pane.
- To enable/disable individual debug points: (i) click its circle indicator, or (ii) right-click the debug point and toggle the **Enabled** command on/off.
- To enable/disable all debug points, click the respective tool bar icon.
- To remove individual debug points: (i) right-click the debug point and select **Delete**, or (ii) select the debug point and click the toolbar icon **Clear This Debug Point**.
- To remove all debug points, click the toolbar icon **Clear All Debug Points**.
- You can enter a break condition, which is an XPath expression that must evaluate to `true()` for the breakpoint to be enabled. To set a break condition for a given breakpoint, double-click *Enter break condition* in that breakpoint's right-hand column. Alternatively, right-click the breakpoint and select **Edit**. The second breakpoint in the screenshot above has a break condition. Note that break conditions are available for breakpoints only, not for tracepoints.

## 5.7 Page Sources Pane

The **Page Sources Pane** (*screenshot below; see User Interface*[251] *for its location*) is where page sources are managed. These page sources provide both (i) the structure of data trees used in the design, as well as (ii) the data held in the data trees.



```
Page Sources
Sources
  Namespaces
  $PERSISTENT
  $XML1
    Default File: ...
    Load on first use, XPath Context
    root
      DesiredOffice  ="if ( count($DB1/DB/RowSet/Row) > 0 ) then $DB1/DB/RowSet/Row[1]/@id else """"
      DesiredYear  ="min(distinct-values($DB2/DB/RowSet/Row[@Office=$XML1/root/@DesiredOffice]/@Year))"
  $DB1  OfficeSales_DB (shared with 2 other page(s))
    Load on first use
    DB
      RowSet
        Row
          id Read Only, Primary Key ="(: calculate a new unique id as the db doesn't generate one for us :)
                                      let $all := $DB1/DB/RowSet/Row/@id
                                      let $ids := remove($all, index-of($all, ""))
                                      let $id := if (empty($ids)) then 1 else max($ids) + 1
                                      return $id"
          City
      OriginalRowSet
  $DB2  OfficeSales_DB1 (shared with 1 other page(s))
    Load on first use
```

The pane provides the following main functionality:

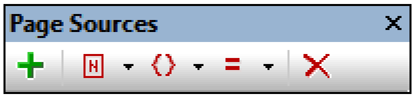
- Displays all the page sources of the page currently selected in the Pages Pane[257]
- Displays all the namespaces declared for the active project
- Enables page sources to be added to the page, via the toolbar's **Add Source** icon
- Enables namespaces to be added to the project, via the toolbar's **Add Namespace** icon
- Enables the default data file of a page source to be set. The data file provides the data that goes into the nodes of the page source
- Enables elements and attributes to be added to a tree, relative to the selected node
- Enables elements and attributes to be given fixed values or XPath-generated values on page load
- Enables the default XPath context node to be set (*the highlighted node in the screenshot above*). The selected node will be the context node for all XPath expressions defined for that page
- Enables nodes to be associated with controls in the page design. This is done by dragging the node onto the control. The associated node (called the page source link) is displayed in bold. When you hover over such a node in the page source tree, a popup provides information about the associated

control/s in the design. Controls that are associated with a page source link have an icon at the control's top left. Hovering over the icon displays information about the associated page source link.
- Enables items in the pane (namespaces, trees, elements, and attributes) to be deleted

For detailed information about how to manage and work with page sources, see the section, Page Sources (Data Sources)[315].

## Manually creating a tree structure

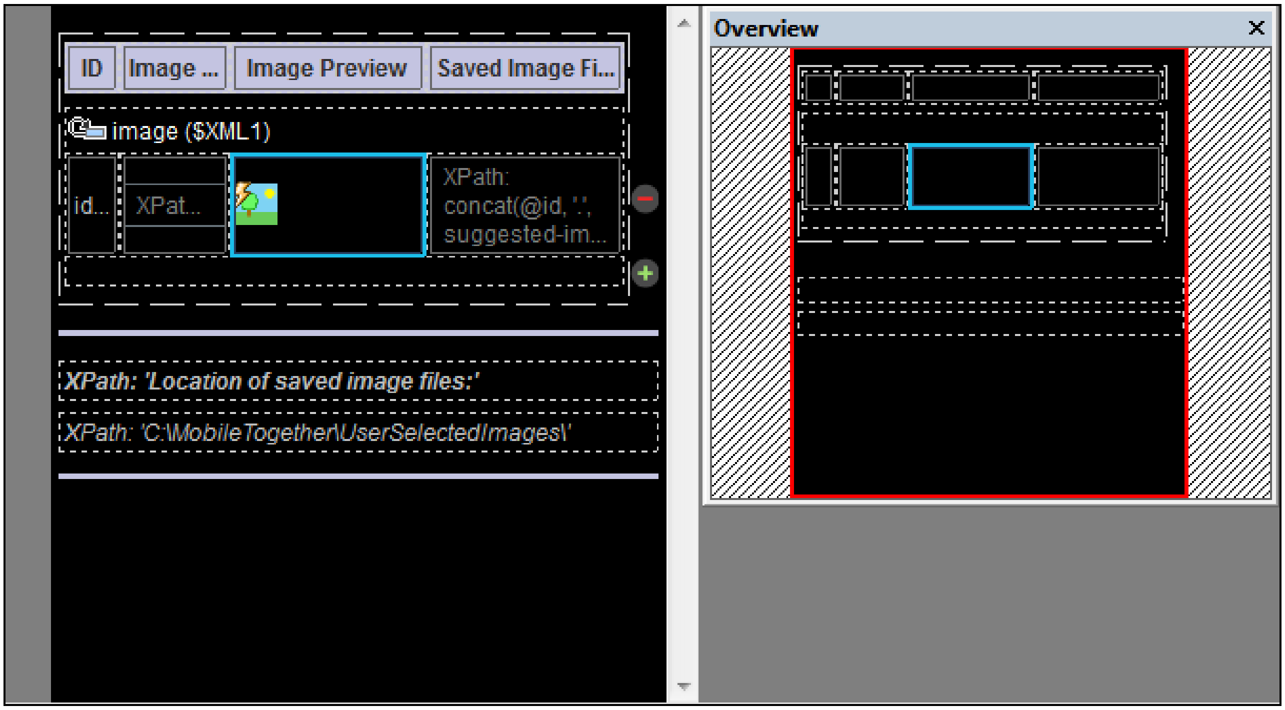
Elements and attributes can be added relative to any node in a tree structure (including the root node[349]), and they can be deleted. Select a node in a page source, and click the appropriate toolbar command (*see toolbar screenshot below*). Temporary elements and attributes are intended to hold data used for calculations or data that for any other reason should not be saved to file. The data of temporary nodes is not saved.



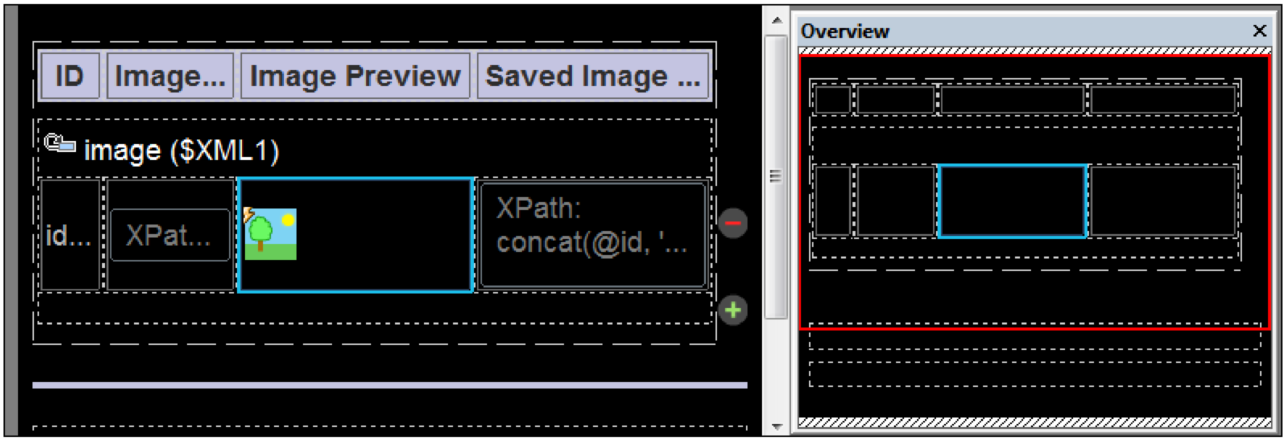
| Icon | Command | Does this... |
|---|---|---|
| + | **Add Source** | Displays the Add Page Source dialog[317]. A root node[349] is created for the page source that is added. Only one child element can be added to a root node. |
| N ▾ | **Add Namespace** | Inserts or appends a namespace declaration under the *Namespace* entry. Edit the default prefix if you want, and enter a namespace. |
| {} ▾ | **Add Element** | Inserts, appends, or adds a child element relative to the selected node. |
| = ▾ | **Add Attribute** | Inserts, appends, or adds a child attribute relative to the selected node. |
| ✕ | **Delete** | Deletes the selected node. |

# 5.8 Overview Pane

The **Overview Pane** (*screenshot below; see User Interface*[251] *for its location*) shows a small version of the Page Design View[253]. To see the default location of the Overview Pane, go to The Graphical User Interface (GUI)[251].



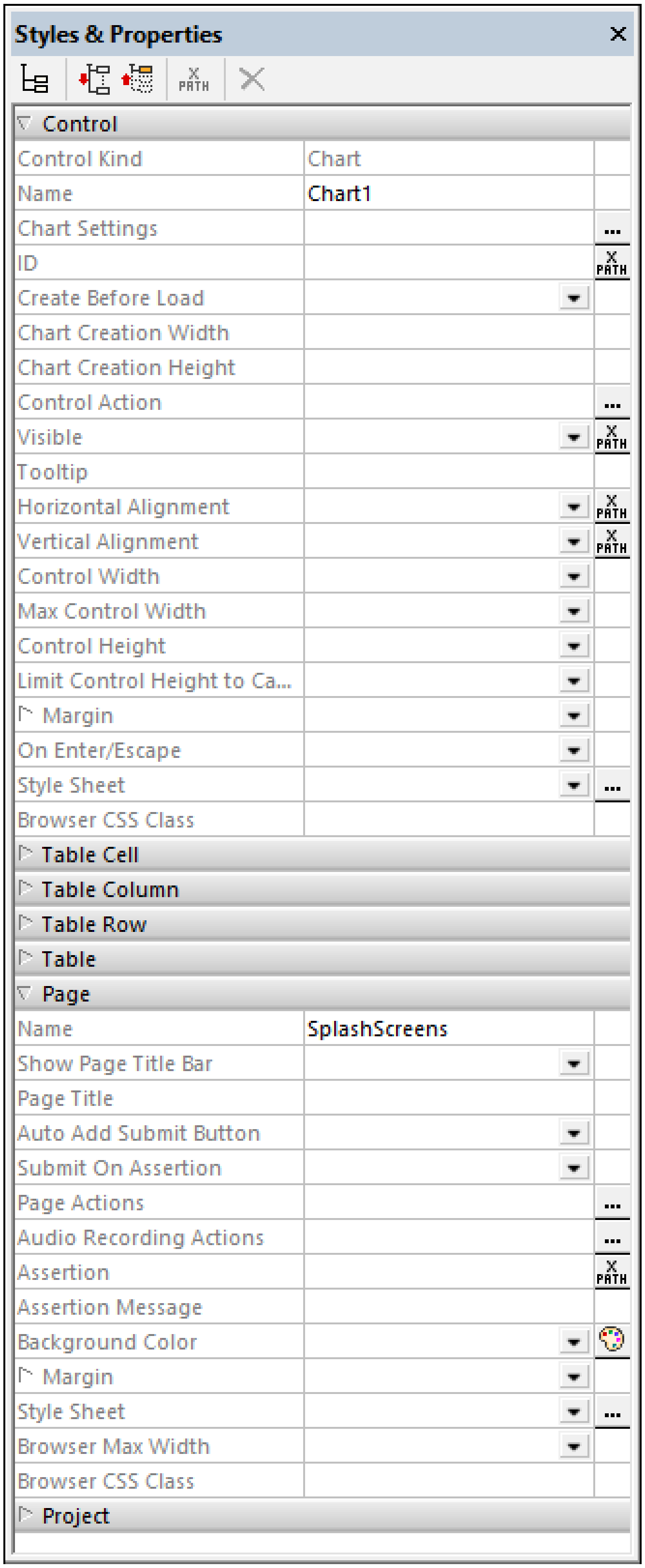
In the Overview Pane in the screenshot above, the extent of the design is indicated by the black area. The red rectangle is the viewport. It indicates the part of the design that is currently visible in Page Design View[253], and its dimensions correspond to those of the displayed part of the design (*compare the screenshots above and below*). If a component is selected in the design, the selected component will be highlighted in the Overview Pane (*see screenshots*). If part of the design is not displayed in the viewport, you can grab the red rectangle in the Overview Pane and move it so that the required part of the design is brought within the viewport. The screenshot above shows that the entire design is inside the viewport, but in the screenshot below, the lower parts of the design are outside the viewport. These parts can be moved into view by dragging the red rectangle down.



The Overview Pane is useful for managing the viewing of large diagrams.

## 5.9 Styles & Properties Pane

The **Styles & Properties Pane** (*screenshot below; see User Interface*[251] *for its location*) displays the properties of the currently selected page controls, pages, and project, and enables these to be conveniently edited in one place. The Styles & Properties Pane is divided into the following sub-panes: (i) *Control,* for the properties of the page control currently selected in the design, (ii) *Page,* for the properties of the current page, and (iii) *Project,* for the properties of the current project. The Table control and its parts (cells, columns, and rows) are each given a separate pane (*see screenshot below*). Table and table-part sub panes are only displayed if, in the design, a part of a table is selected.

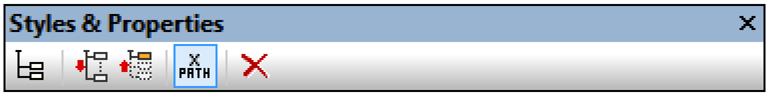


Styles & Properties

| | |
|---|---|
| ▽ Control | |
| Control Kind | Chart |
| Name | Chart1 |
| Chart Settings | |
| ID | |
| Create Before Load | |
| Chart Creation Width | |
| Chart Creation Height | |
| Control Action | |
| Visible | |
| Tooltip | |
| Horizontal Alignment | |
| Vertical Alignment | |
| Control Width | |
| Max Control Width | |
| Control Height | |
| Limit Control Height to Ca... | |
| Margin | |
| On Enter/Escape | |
| Style Sheet | |
| Browser CSS Class | |
| ▷ Table Cell | |
| ▷ Table Column | |
| ▷ Table Row | |
| ▷ Table | |
| ▽ Page | |
| Name | SplashScreens |
| Show Page Title Bar | |
| Page Title | |
| Auto Add Submit Button | |
| Submit On Assertion | |
| Page Actions | |
| Audio Recording Actions | |
| Assertion | |
| Assertion Message | |
| Background Color | |
| Margin | |
| Style Sheet | |
| Browser Max Width | |
| Browser CSS Class | |
| ▷ Project | |

## Styles & Properties Pane toolbar

The commands available in the Styles & Properties toolbar (*screenshot below*) are listed in the table below:



| Icon | Command | Does this... |
|---|---|---|
| | **List Non-Empty** | Toggles between displaying all properties and only those that have values defined for them. Properties that have default values are considered empty. |
| | **Expand All** | Expands all sub-panes. |
| | **Collapse All** | Collapses all sub-panes. |
| X PATH | **Edit XPath** | Enabled when a property that requires an XPath expression is selected. It displays the XPath/XQuery Window. |
| ✕ | **Reset** | Resets the property to empty or to its default value. |

## Entering and editing property values

Property values are entered or edited depending on the ways that are available to enter the value. These ways are listed in the table below. Often, more than one way is available to enter a value, enabling you to chose whatever way you prefer. If it is possible to enter a value by way of an XPath expression, then the property will have an XPath icon in the third column, or the XPath icon in the toolbar will be enabled when the property is selected.

| Entry method | Do this... |
|---|---|
| **Text field** | Double-click, and enter or edit the property value. |
| **Combo box** | Select a value from the dropdown list. Alternatively, enter or edit a value. |
| **Color palette** | Click to pop up the color palette, then select a color. |
| **Edit XPath** | Click to pop up the Edit XPath/XQuery Expression dialog, then enter an XPath/XQuery expression. The expression provides the value of the property. |
| **Additional dialog** | Click to open an additional dialog that will enable a value to be entered. This dialog could be, for example, the Actions dialog or the Open File dialog. |

## Context menu

The context menu of a property provides a quick way to access the additional functionality listed below.

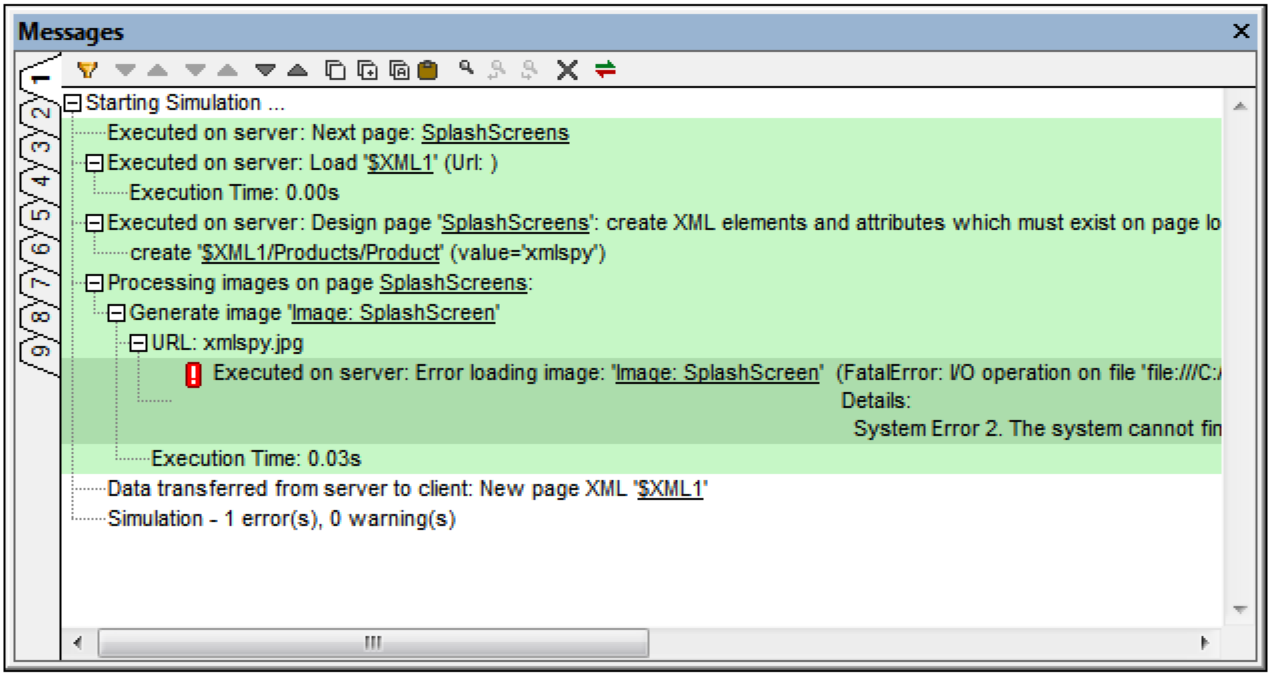
- Set a fixed value from a list of valid pre-selected values.
- Set an XPath expression that will, at run time, evaluate to the value of the property.
- Set a platform default. This option is enabled if the default value is variable across platforms.

- Reset to the default value of the property. The default value of a property can be seen if you place the cursor over the name of the property.
- Cut, Copy, Paste commands.
- Access the MobileTogether Designer Style Sheets[1318] dialog, in which styles can be managed at various levels (from document-wide to a single control type).
- List, in the Listings Pane[281], controls that have the same direct style value as that of the selected property. A direct style value refers to a value that is entered directly at the control level—as opposed to a value that is set via a style sheet[1318].
- List, in the Listings Pane[281], controls that have the same style value as that of the selected property, whether the value is entered directly at the control level or is set via a style sheet[1318].
- Group, in the Listings Pane[281], controls by their direct style value. All controls that use the selected property will be listed, organized into groups according to the value that has been set for the selected property. For example, if this command is executed on the **`Text Size`** property, then controls are listed in groups for different text sizes (large, medium, 10pt, etc). Direct style value refers to a value that is entered directly at the control level—as opposed to a value that is set via a style sheet[1318].
- Group, in the Listings Pane[281], controls by their style value, whether the value is entered directly at the control level or is set via a style sheet[1318]. The effect is similar to that of the previous command; it groups controls on the basis of property values.

# 5.10 Messages Pane

The **Messages Pane** (*screenshot below, see User Interface*[251] *for its location*) displays messages in the following contexts:

- It reports the validation results[1589] of the currently active project. This includes all pages in the project. If an error is reported, the error message contains a link that points to the component that generated the error.
- During simulations[1355], it provides a detailed and step-by-step report of the progress of the workflow.



```
Starting Simulation ...
    Executed on server: Next page: SplashScreens
    Executed on server: Load '$XML1' (Url: )
        Execution Time: 0.00s
    Executed on server: Design page 'SplashScreens': create XML elements and attributes which must exist on page lo
        create '$XML1/Products/Product' (value='xmlspy')
    Processing images on page SplashScreens:
        Generate image 'Image: SplashScreen'
            URL: xmlspy.jpg
                Executed on server: Error loading image: 'Image: SplashScreen'  (FatalError: I/O operation on file 'file:///C:/
                                                                                 Details:
                                                                                  System Error 2. The system cannot fin
            Execution Time: 0.03s
    Data transferred from server to client: New page XML '$XML1'
    Simulation - 1 error(s), 0 warning(s)
```

The toolbar of the Messages Pane contains commands that enable the following actions: filtering of the messages, navigation of the messages, copying of messages, pasting of the server log message, searching the messages, clearing the current tab, and setting the background colors of server and client log messages. There are nine Messages tabs. Therefore, you can retain the results of a validation in one tab while carrying out a simulation with a new tab open.

*Paste server log messages to locate error sources in the design*

If, when a solution runs on MobileTogether Server, the server logs show an error, you can hover over the error message to display a **Copy** button that enables you to copy the error message to the clipboard.

Now if you open the design of the solution in MobileTogether Designer, you can paste the error message in the Messages Pane. Do this by clicking the **Paste Server Log** icon of the pane's toolbar (*see screenshot above*). The server log message that you copied to the clipboard will be pasted. From details in the message, MobileTogether Designer will locate the referenced points in the design, and the message will be displayed with links to these design definitions. Click the link/s to find the source of the error.

*Using XPath trace() to generate messages*

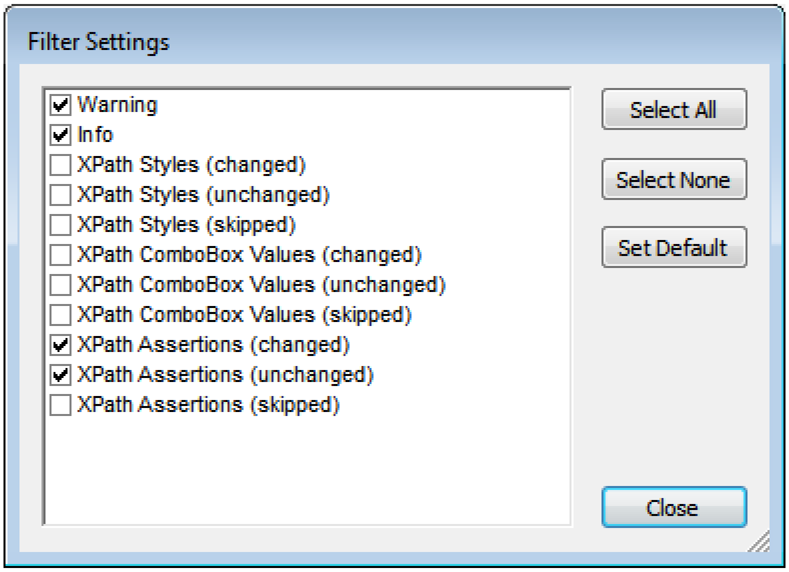
You can use XPath's **trace()** function to generate messages in the Messages Pane. For example, the following XPath expression will generate a message that reflects the value in the **Company/Name** node:

```
if (Company/Name="Altova") then trace("Company is Altova") else
trace("Company is not Altova")
```

The text of the relevant **trace** function is returned as the message in the Messages Pane. This message will also be a link, which when clicked takes you to the the originating **trace** function.

*Filtering messages*

You can specify what kind of messages are displayed in the Messages Pane. To do this, click the **Filter** button in the toolbar of the Messages Pane (*screenshot above*). This displays the Filter Settings dialog (*screenshot below*). Select the message types you want to display and click **Close**. This feature can be very useful, for example, if there are too many messages and you wish to focus on just one type of message.

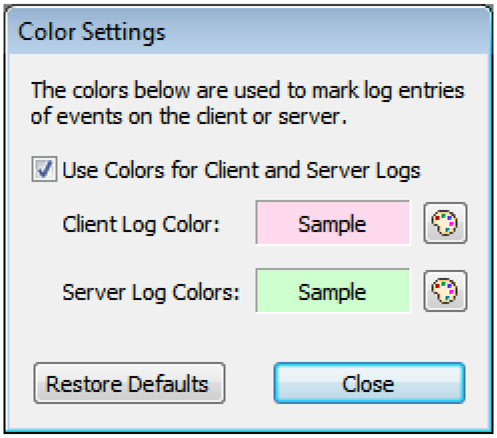


Filter Settings

☑ Warning
☑ Info
□ XPath Styles (changed)
□ XPath Styles (unchanged)
□ XPath Styles (skipped)
□ XPath ComboBox Values (changed)
□ XPath ComboBox Values (unchanged)
□ XPath ComboBox Values (skipped)
☑ XPath Assertions (changed)
☑ XPath Assertions (unchanged)
□ XPath Assertions (skipped)

*Color settings*

For messages that are displayed during simulations, different colors can be set for actions that take place on the server and on the client. If you set clearly distinguishable colors, you can very easily follow the workflow in the Messages Pane. This can be of great help in debugging. To set customs colors, click the **Colors** button in the toolbar of the Messages Pane (*screenshot above*). This displays the Color Settings dialog (*screenshot below*), in which you can set the colors you want.



Color Settings

The colors below are used to mark log entries of events on the client or server.

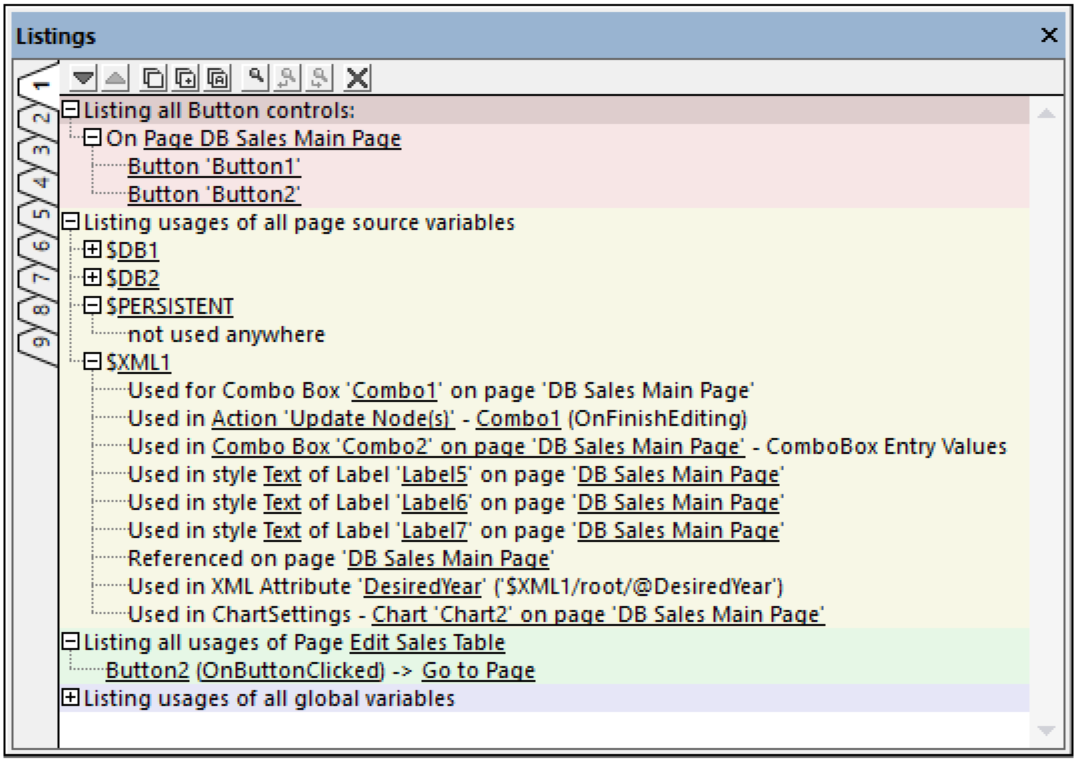
☑ Use Colors for Client and Server Logs

Client Log Color: Sample

Server Log Colors: Sample

# 5.11 Listings Pane

The **Listings Pane** (*screenshot below, see* User Interface[251] *for its location*) displays the output of listing and grouping commands, such as menu command **Refactor | List Usages of All Global Variables**[1610].



```
⊟ Listing all Button controls:
  ⊟ On Page DB Sales Main Page
      Button 'Button1'
      Button 'Button2'
⊟ Listing usages of all page source variables
  ⊞ $DB1
  ⊞ $DB2
  ⊟ $PERSISTENT
      not used anywhere
  ⊟ $XML1
      Used for Combo Box 'Combo1' on page 'DB Sales Main Page'
      Used in Action 'Update Node(s)' - Combo1 (OnFinishEditing)
      Used in Combo Box 'Combo2' on page 'DB Sales Main Page' - ComboBox Entry Values
      Used in style Text of Label 'Label5' on page 'DB Sales Main Page'
      Used in style Text of Label 'Label6' on page 'DB Sales Main Page'
      Used in style Text of Label 'Label7' on page 'DB Sales Main Page'
      Referenced on page 'DB Sales Main Page'
      Used in XML Attribute 'DesiredYear' ('$XML1/root/@DesiredYear')
      Used in ChartSettings - Chart 'Chart2' on page 'DB Sales Main Page'
⊟ Listing all usages of Page Edit Sales Table
    Button2 (OnButtonClicked) -> Go to Page
⊞ Listing usages of all global variables
```

Note the following points:

- The pane has nine tabs, so you can leave the results in one tab and switch to a new tab. This can be useful, for example, if you want to compare lists.
- The window is not cleared each time a new list is displayed, but each list is displayed in a new color *(see screenshot above)*.
- Lists and their components can be expanded and collapsed.
- Components in the list are hyperlinked. So you can immediately go to a component, or to the dialog where a component is defined, by clicking that component in the list.
- The toolbar of the Listings Pane contains commands that enable the following: navigation of the lists, copying of list items, searching the lists, and clearing the tab.

## Available listing and grouping commands

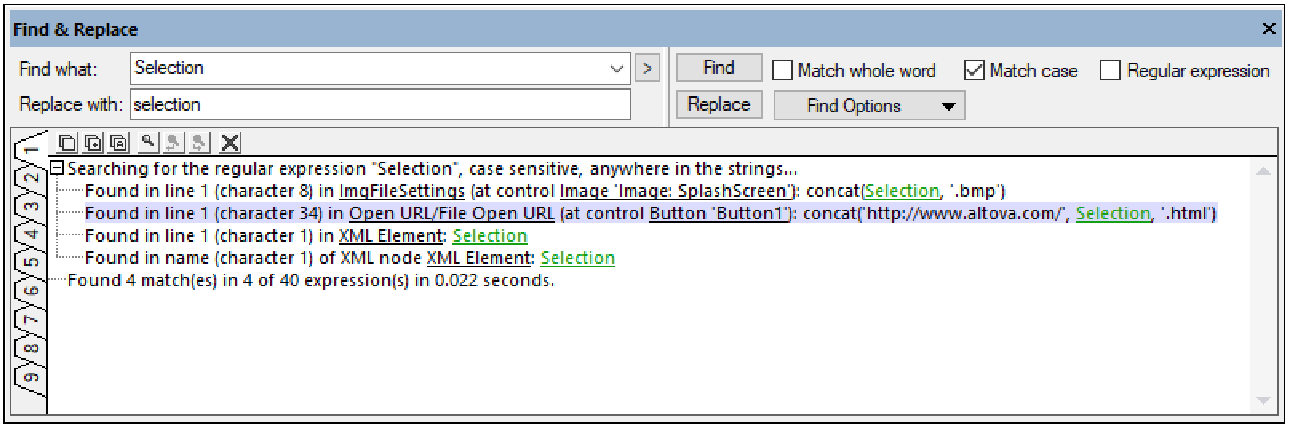
Lists generated by the following commands are displayed in the Listings Pane.

- List All Controls Of Kind[266]
- Group Controls By Style Value[1328]
- Group Controls By Direct Style Value[1328]
- List Controls With Same Style Value[1328]
- List Controls With Same Direct Style Value[1328]

- List Usages of All Global Variables 1610
- List Usages of All Page Source Variables 1610
- List Page Sources by Attribute 1610
- List Usages of All User-Defined XPath/XQuery Functions 1611
- List Usages of All Action Groups 1611
- List Usages of All Stylesheets 1612
- List All File and Directory References 1612
- List All External Data References 1612
- List Unused Functions, Variables, Etc 1613
- List Text Size Auto Fit Groups 1636
- List All Usages of an Action Kind 667
- List All Page Usages 257

# 5.12 Find & Replace Pane

The **Find & Replace Pane** (*screenshot below, see User Interface* 251 *for its location*) enables you to search for text strings in all pages of design or in the current page. The search scope includes XPath expressions and XPath function names, variable names and values, control names and properties, and actions and action groups. You can also replace all instances of the found text string and use regular expression to find text.



Find what: Selection
Replace with: selection
☐ Match whole word ☑ Match case ☐ Regular expression

```
⊟ Searching for the regular expression "Selection", case sensitive, anywhere in the strings...
    Found in line 1 (character 8) in ImgFileSettings (at control Image 'Image: SplashScreen'): concat(Selection, '.bmp')
    Found in line 1 (character 34) in Open URL/File Open URL (at control Button 'Button1'): concat('http://www.altova.com/', Selection, '.html')
    Found in line 1 (character 1) in XML Element: Selection
    Found in name (character 1) of XML node XML Element: Selection
  Found 4 match(es) in 4 of 40 expression(s) in 0.022 seconds.
```

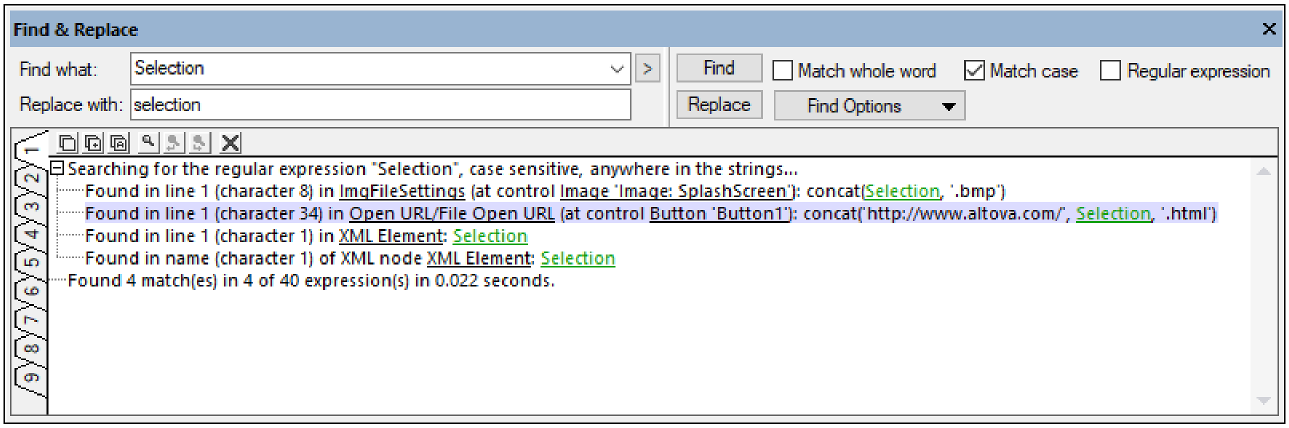
## Searching for text

To search for text in the design, do the following:

1. In the *Find What* entry field (*see screenshot above*), enter the string to find, or use the dropdown list to select a string from one of the last 10 strings. You can also use regular expressions (explained below 284) to find text.
2. Optionally, narrow or expand the search by, respectively, selecting or deselecting the *Match whole word* and/or *Match case* options. If *Match whole word* is selected, then only the exact words in the text will be matched. For example, if the search term is `fit`, then only the word `fit` will be matched; the `fit` in `fitness`, for example, will not be matched.
3. Select the scope of the search by toggling on/off the options in the *Find Options* combo box. You can restrict the search to the current page, and include XPath functions and/or action groups in the search.
4. Click **Find** or press **Enter**.

## Results of the search

There are nine *Result* tabs (*see screenshot below*). Results are displayed in the currently selected *Result* tab (and overwrite any previous results in that tab). You can switch to a new tab if you want to keep a previous result.



Find what: Selection

Replace with: selection

□ Match whole word ☑ Match case □ Regular expression

```
⊟ Searching for the regular expression "Selection", case sensitive, anywhere in the strings...
    Found in line 1 (character 8) in ImgFileSettings (at control Image 'Image: SplashScreen'): concat(Selection, '.bmp')
    Found in line 1 (character 34) in Open URL/File Open URL (at control Button 'Button1'): concat('http://www.altova.com/', Selection, '.html')
    Found in line 1 (character 1) in XML Element: Selection
    Found in name (character 1) of XML node XML Element: Selection
  Found 4 match(es) in 4 of 40 expression(s) in 0.022 seconds.
```

Result tabs provide the following functionality:

- Each instance of a found search term is displayed on a separate line and contains links to the relevant design objects. For example, in the screenshot above, the instance that is highlighted refers to an instance of the search term *Selection* in an XPath expression that is defined for an Open URL action[681] of a Button control[417]. Clicking the three links take you, respectively, from left, to the action definition, the button control in the design, the XPath expression of the action.
- You can use the *Copy* icons in the toolbar of the *Results* tab (*see screenshot above*) to copy the following to the clipboard: (i) the selected result, (ii) the selected result and its children, (iii) all results.
- The *Results* tab has its own Find function, which enables you to search the results for a term. Click the **Find** icon in the toolbar to do this. Navigate these results with the **Find Previous** and **Find Next** icons.
- The **Clear** icon of the *Results* tab toolbar clears all the results in the current tab.

*Replacing text*

To replace the search term in the found instances with another text string, enter the new string in the *Replace With* text box and click **Replace**. Note that a Replace action might not work if the replaced text causes invalidity, for example, when a replacement in an XPath expression invalidates the expression, or when a replaced style value is not a valid value.

## Using regular expressions

You can use regular expressions (regex) to find a text string. To do this, first, switch the *Regular expression* option on (see *Searching for text*[283] above). This specifies that the text in the search term field is to be evaluated as a regular expression. Next, enter the regular expression in the search term field. For a brief description of regular expression metacharacters, see the section *Regular expression metacharacters*[284] below.

*Regular expression metacharacters*

Given below is a list of regular expression metacharacters.

| | |
|---|---|
| . | Matches any character. This is a placeholder for a single character. |
| ( | Marks the start of a tagged expression. |
| ) | Marks the end of a tagged expression. |
| `(abc)` | The `(` and `)` metacharacters mark the start and end of a tagged expression. Tagged |

| | |
|---|---|
| | expressions may be useful when you need to tag ("remember") a matched region for the purpose of referring to it later (back-reference). Up to nine expressions can be tagged (and then back-referenced later, either in the Find or Replace field).<br>For example, `(the) \1` matches the string `the the`. This expression can be literally explained as follows: match the string "the" (and remember it as a tagged region), followed by a space character, followed by a back-reference to the tagged region matched previously. |
| `\n` | Where `n` is a **`variable`** that can take integer values from `1` through `9`. The expression refers to the first through ninth tagged region when replacing. For example, if the find string is `Fred([1-9])XXX` and the replace string is `Sam`**`\1`**`YYY`, this means that in the find string there is one tagged expression that is (implicitly) indexed with the number **`1`**; in the replace string, the tagged expression is referenced with **`\1`**. If the find-replace command is applied to `Fred2XXX`, it would generate `Sam2YYY`. |
| `\<` | Matches the start of a word. |
| `\>` | Matches the end of a word. |
| `\x` | Allows you to use a character `x`, which would otherwise have a special meaning. For example, `\[` would be interpreted as `[` and not as the start of a character set. |
| `[...]` | Indicates a *set of characters*. For example, `[abc]` means any of the characters `a`, `b` or `c`. You can also use ranges: for example `[a-z]` for any lower case character. |
| `[^...]` | The complement of the characters in the set. For example, `[^A-Za-z]` means any character except an alphabetic character. |
| `^` | Matches the start of a line (unless used inside a set, *see above*). |
| `$` | Matches the end of a line. Example: `A+$` to find one or more `A`'s at end of line. |
| `*` | Matches 0 or more times. For example, `Sa*m` matches `Sm`, `Sam`, `Saam`, `Saaam` and so on. |
| `+` | Matches 1 or more times. For example, `Sa+m` matches `Sam`, `Saam`, `Saaam` and so on. |

*Representation of special characters*

Note the following expressions.

| | |
|---|---|
| **`\r`** | Carriage Return (CR). You can use either CR (**`\r`**) or LF (**`\n`**) to find or create a new line |
| **`\n`** | Line Feed (LF). You can use either CR (**`\r`**) or LF (**`\n`**) to find or create a new line |
| **`\t`** | Tab character |
| **`\\`** | Use this to escape characters that appear in regex expression, for example: **`\\\n`** |

# 6 Project

This section describes options that are relevant at the project level:

# 6.1 Client-Server Interaction

This topic describes features and settings that determine the level of interaction between the client device and MobileTogether Server.

## MobileTogether Client app or MobileTogether AppStore App

The first decision to take is whether the project should be distributed to MobileTogether Server as a solution for the **MobileTogether Client app** or a solution for a **MobileTogether AppStore App**.

- A MobileTogether Client app is downloaded from an app store by the end user. On the end user device, MobileTogether Client is configured to access one or more MobileTogether Servers. Depending on security considerations, access to a server can be either anonymous or via user login with a password. The end user must be informed about the server configuration and access details. Once the end user has been granted access to a folder on the server, MobileTogether projects that have been deployed as solutions to this folder become accessible to the end user. Access rights to a folder are managed by the administrator of MobileTogether Server. See the MobileTogether Server documentation for details.
- A MobileTogether AppStore App on the other hand is a standalone app that is dedicated to a single solution located on the server. An AppStore App is downloaded from an app store and is started directly on the end user's device. There is no need for MobileTogether Client to be installed in order to run this kind of app. However, contact must be made with the appropriate MobileTogether Server in order to access the solution. The app contains a key that serves as a "handshake" with the solution on the server. Interaction with the server thereafter depends on the value of the *Server Access* setting. For more information, see AppStore Apps[1471].

## A project's Server Access setting

A project's *Server Access*[296] setting specifies the level of server access while the solution runs. There are three options: `Always`, `On Demand`, and `Never`. The default is `always`. The appropriate option should be selected depending upon the kind of access to resources on the server that the solution needs. If the `Never` option is selected, then, after the initial connection to the server has been made, the server will not be accessed any more. For a detailed description of the setting, see Project Properties[296].

## Anonymous login for MobileTogether Client

When a MobileTogether Client app connects to a MobileTogether Server, the end user can log in to the server as a recognized user or anonymously. To log in as a user, a user name and password that is recognized by MobileTogether Server must be used. Alternatively, MobileTogether Server can be configured by the server administrator to grant anonymous access to folders individually. See the MobileTogether Server documentation for details.

## Updating server settings on client devices

In order for a client device to run a solution, the server's access settings must be configured on that device. If the server settings change—for example, if the MobileTogether Server is moved to another machine that has a different IP address—then the server settings on client devices must be modified accordingly. The MobileTogether function `mt-server-config-url`[1262] generates a URL that contains the new server settings and looks something like this: `mobiletogether://mt/change-settings?settings=<json encoded settings>`. This URL can be sent as an email link to the MobileTogether Client device. When the link is tapped, server settings on the client are automatically updated.

The JSON-encoded server settings that are contained in the URL are provided by the argument of the `mt-server-config-url`1262 function (described here1262). For an example of how to use this function, see the example solution **`ClientConfiguration.mtd`** in the `MobileTogetherExamples/SimpleApps` folder of your MobileTogether Designer installation.

**Note:** Links to update server settings do not work in Gmail and some other email applications, but they work in popular clients such as AquaMail, K9, and MailWise. They have been tested in AquaMail and K9 and work correctly in these applications.

## 6.2 Location of Project Files

The project file (`.mtd` file) is the solution file. It is deployed to the server and accessed from there by the MobileTogether Client app. When the project file is deployed to the server it is stored in the server's database of solutions, and the server will reference the file by its name. The project file uses other files (such as XML files and image files), from which it reads data and to which it can write data. These associated files can be stored at the following locations:

⊟ *Deployed to the server with the project file*

- These data files will be read-only on the server.
- The files are stored in the server's database. The advantage is that access to the data files is internally handled within the project.
- A file can be referenced in the design with a relative or absolute path. The filepath in the design (relative or absolute) will be used as the internal file reference on the server database.
- For the details of deployment, see: Deploying the Project 291, Files Pane 259, Deploy to MobileTogether Server 1574.

⊟ *Embedded within the project file*

- This applies to XML data files (typically the default files of page sources).
- The advantage of embedding is that the XML data and images will be transported together with the project file, and will be correctly accessed from within the project. Therefore, server access is not required in order to access these files.
- As with deployed files, embedded XML files will be read-only.
- The disadvantage is that the size of the project file increases.
- To embed a file, right-click its root node 349 in the Page Sources Pane 270, and select **Embed XML in Design File** 359. You can select whether files are automatically re-embedded 296 when the user starts a simulation or deploys the solution to the server.

⊟ *A directory on the server*

- Files of any type can be stored in any directory on the server. These files can be read-write.
- However, care must be taken to correctly configure (i) the file's location when it is added 355, and (ii) MobileTogether Server's *Server Side Solution's Working Directory* setting.
- If files are referenced by relative paths, the relative paths are resolved relative to the Working Directory.
- If files are referenced by absolute paths, the directory containing the file must be a descendant directory of the Working Directory. For example:
  If the absolute path of the referenced file is: `C:\Altova\MobileTogether\Test\First.xml`
  and the Working Directory is set to: `C:\Altova\MobileTogether`
  then the XML file will be accessed correctly.
  It will not be accessed correctly if the Working Directory is set to: `C:\Altova\MobileTogether\Files` or to `C:\Altova\MTD`
- The advantage of using directories on the server to store data files is that the data accessed by the solution will always be up-to-date.

⊟ *A URL accessible over the Internet*

- Files of any type can be stored at any URL that the server can access over the Internet.
- If a file is added as a page source, security authorizations can be set when specifying the properties of the page source.
- The advantages of using Internet locations are: (i) data accessed by the solution will always be up-to-date, and (ii) the solution is portable.

⊟ Also see

Deploy to MobileTogether Server 1574
Files Pane 259
Deploying the Project 291
Data Storage on Servers 312.

# 6.3 Deploying the Project

After you have completed the design of your project in MobileTogether Designer, the project (or design) is ready to be deployed to one or more MobileTogether Servers. In order to deploy the project to a MobileTogether Server, you need to have an HTTP connection to the machine on which the targeted MobileTogether Server is running. Once the project is deployed, it is available as a MobileTogether solution that MobileTogether Client applications running on mobile devices can access.

**Note:** If you try to deploy a solution containing Advanced Edition features to MobileTogether Server Standard Edition, then an error is reported and the solution will not be deployed.

## Deployment and access control

Access to a solution can be controlled on two levels: (i) accessibility of the server where the solution is deployed; (ii) a client device's access to a server.

*Accessibility of the server*

- Deployment to in-house servers behind a firewall automatically restricts access to internal users, for example, to the employees of a company.
- Deployment to servers that allow external access allow external end users to access MobileTogether solutions, for example, the customers or clients of a company.

*Client access to the server*

For each server a set of users can be defined that have access to that server. Access will be available only to those client devices that submit the appropriate user-name and password. The users of a server and their privileges are defined in settings of MobileTogether Server. See the user manual of MobileTogether Server for details of how to define users, roles, and user privileges.

## Deployed files and their locations

The following files are deployed when the project is deployed with the **File | Deploy to MobileTogether Server**[1574] command:

- The project (`.mtd`) file, which is deployed to the server. This is the solution file that will be accessed by the MobileTogether Client app on the client device.
- Deployable files in the Files Pane[259] that have their check boxes selected. These files will be read-only and typically are image files and data files. They can be deployed to the server or client or both. If a file is deployed to the client, then it is transferred from the server to the client once, when the solution is started. If a file is looked up frequently, then it might be better to deploy it to the client. This would save the time required to transfer it from the server each time the file is called.
- Server side solution files that have been added to the Files Pane[259]. These are files that are required by the solution to be on the server. (In the case of DBs, only file-based DBs, such as SQLite and Access, qualify.) When the project is deployed to the server, files that were added to the project as server side solution files will be deployed to the server side solution's working directory.

**Note:** Audio and video files **cannot** be deployed to MobileTogether Server via the MobileTogether Designer project's Deploy to Server mechanism[291]. You can, however, copy audio/video files manually to the server, although you cannot stream them from there via a URL. If you wish to stream audio/video files that are located on your MobileTogether Server, then do the following: (i) use the Load Binary[815] action to load the binary audio/video data to a page source node; (ii) use the Save Binary[815] action to save the data in this node to a file on the client device; (iii) use audio/video playback actions[1108] to play the file that is now saved on the client device. Alternatively, you can save audio/video files to a web server—instead of saving to MobileTogether Server—and use a URL to stream the audio/video file from the web server.

**Note:** If you are deploying a MobileTogether package—and not a MobileTogether solution—then you do not have to select deployable files. All required resources are contained in the package and will be deployed automatically to the server.

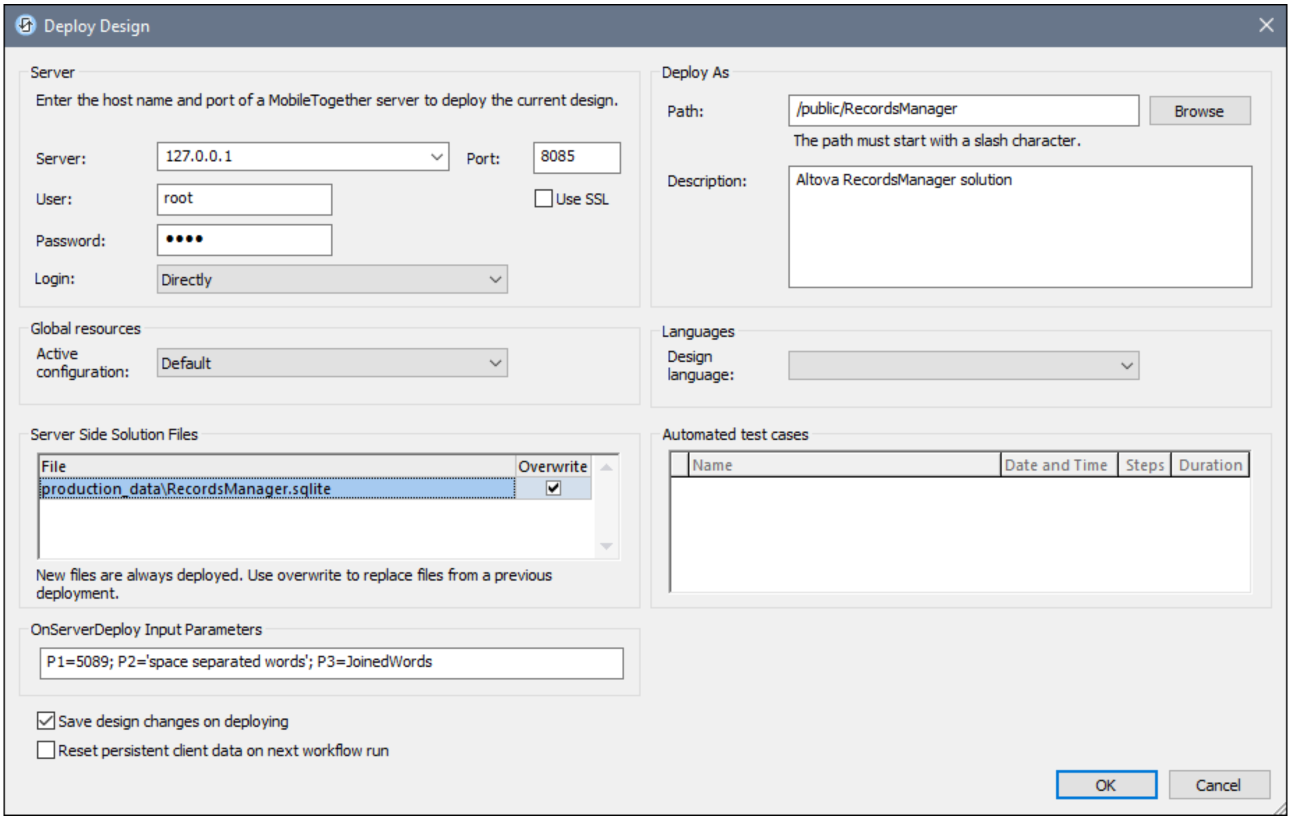
## Deployed files are read-only

Deployed files are read-only. If there is a *Save* action defined for a deployed file, then the design will be invalid—since the deployed file is read-only and cannot be written to. Deployed files should be used to display information or visuals (images, charts, etc) or to read data.

If you want to save data to a file or database, then store the file at a location in the *Server Side Solution's Working Directory* and reference the file correctly in the design. To build a correct reference to the file, (i) specify the file's location when it is added[355], and (ii) make sure that the *Server Side Solution's Working Directory* is correctly set in MobileTogether Server. See the section Location of Project Files | A directory on the server[289] for more information.

## How to deploy a project

A project or a MobileTogether Package[295] is deployed to the server with the **File | Deploy to MobileTogether Server**[1574] command. This command displays the Deploy Design dialog (*screenshot below*), in which you specify the server connection details[1574], whether the server uses SSL communication, and other server deployment settings[1574].



Enter the host name and port of a MobileTogether server to deploy the current design.

Server: 127.0.0.1  Port: 8085

User: root  □ Use SSL

Password: ••••

Login: Directly

Path: /public/RecordsManager

The path must start with a slash character.

Description: Altova RecordsManager solution

Active configuration: Default

Design language:

| File | Overwrite |
|---|---|
| production_data\RecordsManager.sqlite | ☑ |

New files are always deployed. Use overwrite to replace files from a previous deployment.

P1=5089; P2='space separated words'; P3=JoinedWords

☑ Save design changes on deploying

□ Reset persistent client data on next workflow run

**Note:** The *Automated test runs* 1399 pane displays the test cases that have been saved to the design. It appears only if at least one test case has been saved to the design.

## OnServerDeployment event and its input parameters

When a project is deployed, the action tree of the **`OnServerDeployment`** event is executed. You can access this action tree via the More Project Settings dialog of a project 296. The actions of this tree can accept input parameters that will be stored in the **`$MT_InputParameters`** 1300 global variable and can be accessed from there. If a project has one or more **`OnServerDeployment`** actions, then you will be able to enter input parameters when you deploy the project 1574.

## Shortcut links to deployed solutions

In MobileTogether Server, you can create a shortcut link to a deployed solution and place the shortcut link in a container that is not the same as the one containing the solution. This enables a solution to be accessed from different MobileTogether Server containers.

The advantage of this is that by using different input parameters for each shortcut link, you can cause the solution to appear in different ways when opened via different shortcut links. For example, you could create a shortcut link in a container named *Sales* and give it an input parameter `"Department=Sales"`. In the solution, you could specify that when the solution is opened it will be filtered on the value of the department name given in its input parameter. So, when the solution is opened via the shortcut link in the *Sales* container, the solution's records will be filtered to show, because of its input parameter, only the records of the *Sales* department. You could create other shortcut links in other containers (say, *Accounts* or *Legal*), and set corresponding input parameters for them. When the solution is opened via these links, the records that are displayed would be filtered for the corresponding department. The solution that is opened in all these cases would be the original solution. What the shortcut link does is enable you to present and process the solution in different ways according to the input parameters corresponding to the shortcut link.

**Note:** The **`MT_InputParameters`** 1300 global variable of a solution can receive input parameters from various originating points. If this happens, the input parameters are merged. If a key name given at two originating points is the same, then the value defined for shortcuts in MobileTogether Server wins and will be the value that is assigned to the key.

For more information about how to set shortcut links and input parameters in MobileTogether Server, see the MobileTogether Server user manual.

## Updating server settings on client devices

In order for a client device to run a solution, the server's access settings must be configured on that device. If the server settings change—for example, if the MobileTogether Server is moved to another machine that has a different IP address—then the server settings on client devices must be modified accordingly. The MobileTogether function `mt-server-config-url` 1262 generates a URL that contains the new server settings and looks something like this: `mobiletogether://mt/change-settings?settings=<json encoded settings>`. This URL can be sent as an email link to the MobileTogether Client device. When the link is tapped, server settings on the client are automatically updated.

The JSON-encoded server settings that are contained in the URL are provided by the argument of the `mt-server-config-url` 1262 function (described here 1262). For an example of how to use this function, see the example solution **`ClientConfiguration.mtd`** in the `MobileTogetherExamples/SimpleApps` folder of your MobileTogether Designer installation.

**Note:** Links to update server settings do not work in Gmail and some other email applications, but they work in popular clients such as AquaMail, K9, and MailWise. They have been tested in AquaMail and K9 and work correctly in these applications.

⊟ Also see

Deploy to MobileTogether Server 1574, for a description of the Deploy to Server dialog (*screenshot above*)
Location of Project Files 289
Files Pane 259
Data Storage on Servers 312.

# 6.4 MobileTogether Packages

A MobileTogether package is a zip file that is generated from a MobileTogether design. It contains the design in an encrypted form and, optionally, deployable resource files that are used by the design (such as images and CSS stylesheets). The package file has a `.mtp` file extension. A package provides easy portability and the ability to quickly deploy the design (from MobileTogether Designer) as a solution to MobileTogether Server[291].

## Create a package

When you create a package as a `.mtp` file, you have the option to enable the following package privileges:

- To open the package in MobileTogether Designer and deploy the contained design as a solution to MobileTogether Server.
- To open the package in MobileTogether Designer and run a simulation.
- To open the package in MobileTogether Designer and be able to run a simulation as well as deploy the solution to MobileTogether Server.

Note the following points:

- If a solution requires resources, these can be added as server side solution files in the Files Pane[259] and included when the package is created[1572]. When the package is deployed to teh server as a solution these server side solution files will be saved to the server relative to the *Server Side Solution's Working Directory*.
- If a solution requires resources that cannot be included in the package, then these resources must be saved manually to the server. For more information, see Deploying the Project[291] and Export MobileTogether Package[1572].
- After the package has been generated, do not modify it internally or in any other way. Any modification will make the `.mtp` file unopenable in MobileTogether Designer.
- The MobileTogether design which is inside a package cannot be edited when the package is opened in MobileTogether Designer.

How to generate a MobileTogether package is explained in the description of the Export MobileTogether Package[1572] command.

## Open a package

Depending on how the package was created (*see above*), you can do one or both of the following:

- Deploy the design contained in the package as a solution to MobileTogether Server.
- Run a simulation of the design contained in the package.

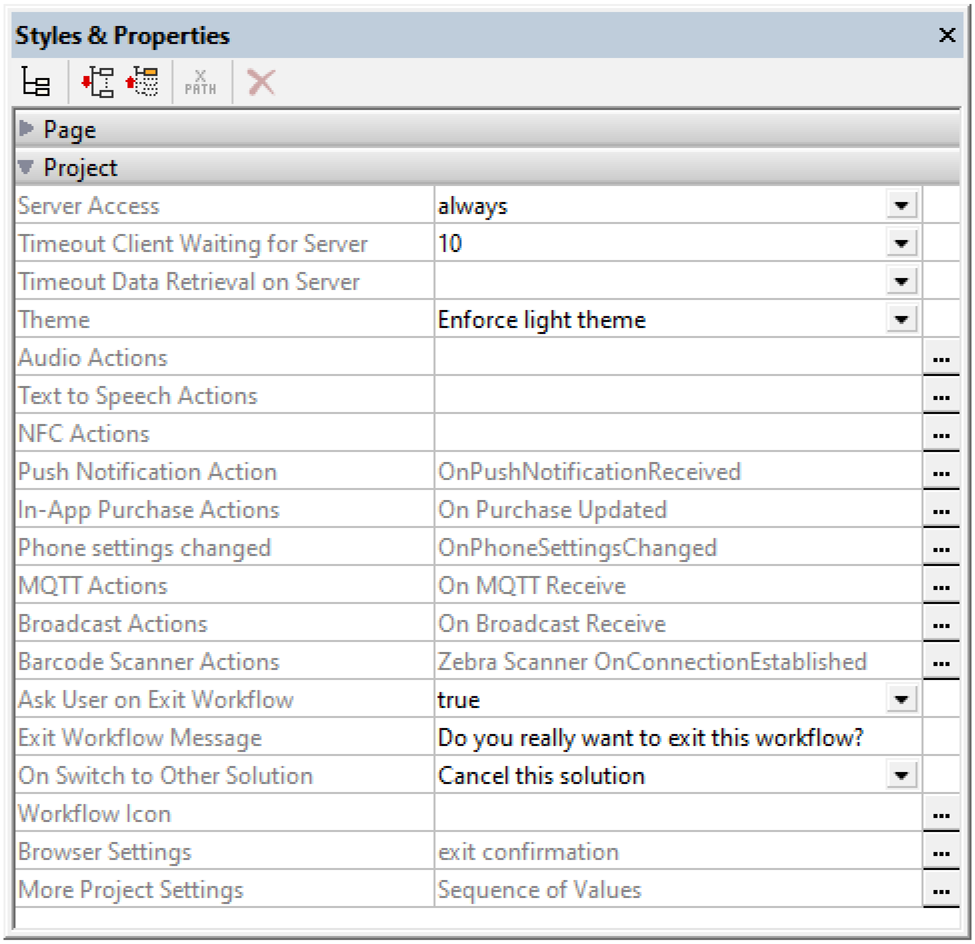
**Note:** When you open a `.mtp` file in MobileTogether Designer, you cannot edit the design that is contained in the package.

## Deploy a package

A MobileTogether package can be opened in MobileTogether Designer and deployed to the server[1574] similarly to the deployment of designs.

## 6.5 Project Properties

Project properties are defined in the Styles & Properties Pane[274] and are described below. If you place your mouse over the property name, a popup appears that contains a brief description of the property.



Styles & Properties

Page

Project

| | |
|---|---|
| Server Access | always |
| Timeout Client Waiting for Server | 10 |
| Timeout Data Retrieval on Server | |
| Theme | Enforce light theme |
| Audio Actions | |
| Text to Speech Actions | |
| NFC Actions | |
| Push Notification Action | OnPushNotificationReceived |
| In-App Purchase Actions | On Purchase Updated |
| Phone settings changed | OnPhoneSettingsChanged |
| MQTT Actions | On MQTT Receive |
| Broadcast Actions | On Broadcast Receive |
| Barcode Scanner Actions | Zebra Scanner OnConnectionEstablished |
| Ask User on Exit Workflow | true |
| Exit Workflow Message | Do you really want to exit this workflow? |
| On Switch to Other Solution | Cancel this solution |
| Workflow Icon | |
| Browser Settings | exit confirmation |
| More Project Settings | Sequence of Values |

### Server Access

This option specifies the level of server access while the solution runs. The default is `always`.

- *Always:* Connection to the server is required in order to run the solution. The server is continually accessed while the solution runs.
- *On Demand:* The MobileTogether Client app runs the solution on its own; it connects to the server only when it needs to exchange data with the server. To run the solution, the app uses data in the internal `$PERSISTENT`[349] tree, other persistent data, or embedded data. You can use the XPath function `mt-has-serveraccess`[1262] to check whether a server connection exists, and then use actions to save appropriately. For example, if no connection exists, then the data can be saved as persistent data on the client. As soon as a connection to the server is established, data can be saved to databases and/or files on the server.
- *Never:* The MobileTogether Client app runs the solution entirely on its own and without needing a connection to the server or any data from the Internet.

▼ Timeout: Client Waiting for Server

The amount of time the client waits for a response from the server. The value is an integer value in seconds that can be entered or selected from the dropdown list of the combo box. The default value is 15 seconds. If the timeout period is exceeded, then an error message is displayed on the client.

▼ Timeout: Data Retrieval on Server

This is the amount of time the server waits for data to be retrieved from a source external to the server (from a DB or URL, for example). The value is an integer value in seconds that can be entered or selected from the dropdown list of the combo box. The default value is 10 seconds. If the timeout period is exceeded, then an error message is displayed. An exception to this is when load actions have the setting *On error* set to `Continue`. In this case, the *On Error* actions of that action's `Continue` setting are executed.

▼ Theme

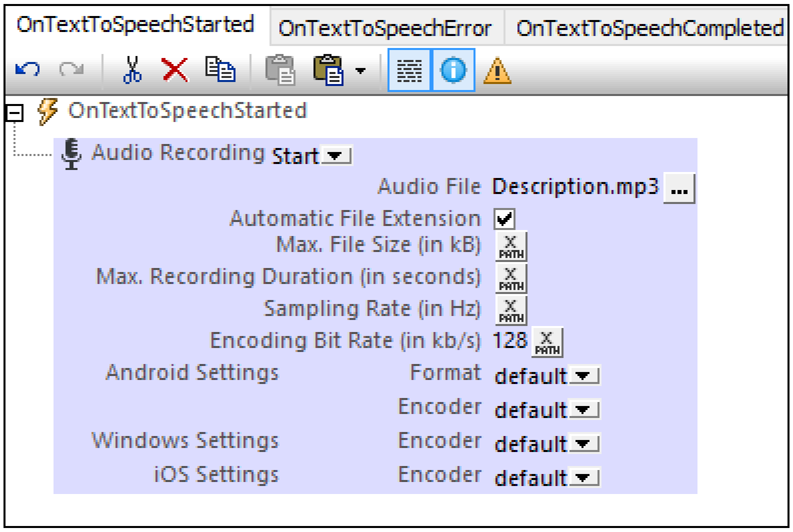
Three options are available: (i) *Use System Settings*, (ii) *Enforce Light Theme*, (iii) *Enforce Dark Theme*. The default is *Use System Settings*. A light theme is one in which dark text is displayed on a light background. A dark theme is one in which light text is displayed on a dark background. If the style definition is *Use System Settings*, then the choice of theme is determined by the user's device.

▼ Audio Actions

Audio events are defined globally for the entire project. Three events are available: `OnAudioStarted`, `OnAudioError` and `OnAudioCompleted`. The actions that are defined for these events **apply to all Audio playback events in the project**. Clicking the property's **Additional Dialog** button displays a dialog containing the definitions of the project's Audio events. For each event, you can define the actions to perform by dragging and dropping actions from the left-hand Actions pane into the event's tab. *For more information, see the description of the Audio (Playback) feature*[1108]*.*

▼ Text to Speech Actions

When you click the **Additional Dialog** button of the property the Text to Speech actions of the Project Actions dialog are displayed (*see screenshot below*).



OnTextToSpeechStarted

Audio Recording Start

| | | |
|---|---|---|
| | Audio File | Description.mp3 ... |
| | Automatic File Extension | ☑ |
| | Max. File Size (in kB) | |
| | Max. Recording Duration (in seconds) | |
| | Sampling Rate (in Hz) | |
| | Encoding Bit Rate (in kb/s) | 128 |
| Android Settings | Format | default |
| | Encoder | default |
| Windows Settings | Encoder | default |
| iOS Settings | Encoder | default |

The following Text to Speech events are available:

- `OnTextToSpeechStarted`: Actions specified in this pane are executed in sequence as soon as playback of a Text to Speech action 727 starts. For example, as shown in the screenshot above, an Audio Recording action 724 can be started to record the Text to Speech playback in a file.
- `OnTextToSpeechError`: Actions to execute if there is a Text to Speech error, such as the text not being found.
- `OnTextToSpeechCompleted`: Actions to execute when a Text to Speech playback is completed. You could, for example, start another Text to Speech playback by specifying a Text to Speech action for this event.

### NFC Actions

Enables actions to be defined for two NFC-related events 1121:

- `OnPushNdefMessageCompleted` specifies what action/s to carry out when the transmission of NFC data (via NFC Push 747) has been completed.
- `OnNfcTagDiscovered` specifies what (additional) action/s to carry out when an NFC tag is discovered 1119.

Click the property's **Additional Dialog** button to go to the definitions of the two events. See NFC-related events 1121 for more information.

### Push Notification Action (OnPushNotificationReceived)

At design time, opens the `OnPushNotificationReceived` event tab, in which you can specify the actions to carry out when a push notification is received. When an action is added to the event, then the `$MT_PUSHNOTIFICATION` page source is automatically added to the design 1128.

When a push notification (PN) is received on a device, one of two alternatives is carried out depending on the *If solution is already running on reception* 753 setting:

- The $MT_PUSHNOTIFICATION page source of the receiving solution is silently updated with the PN's payload, and the actions in the OnPushNotificationReceived event tab are executed. All of this is done directly, without displaying the PN.
- The PN is displayed. When the user taps the PN (or a button in the PN), the following happens: (i) The solution to start is opened if it is not already running; (ii) The solution's $MT_PUSHNOTIFICATION page source is updated with data from the PN's payload; (iii) The actions in the OnPushNotificationReceived event tab are executed.

See Push Notifications 1125 for more information.

▼ In-App Purchase Actions

The In-App Purchase Actions button takes you to the Actions dialog for the OnPurchaseUpdated event. During an in-app purchase, after the end user on the client device confirms a purchase, the app store sends data about the purchase to the device. The data will be stored in a new Purchase element of the **$MT_IN_APP_PURCHASE** 1507 page source. It is when this update to the page source occurs that the OnPurchaseUpdated event is triggered. In the Actions dialog of the OnPurchaseUpdated event, you can define actions that you want to perform at this point in the in-app purchase process. Typically, you would (i) check the response code 1513 of the purchase transaction for success/failure, and (ii) acknowledge the purchase 1511.

See the topics Purchase Products 1511 and Example Project: The "Buy" Button 1524 for more information.

▼ Phone Settings Changed

When a phone setting is changed, this is an event (OnPhoneSettingsChanged), for which you can trigger a set of any of the standard actions. Click the **Additional Options** button at the right of the project property to define the set of actions you want to execute when this event is triggered. The event would be triggered when any phone setting is changed, for example, the theme or the language.

▼ MQTT Actions

In addition to setting actions to perform at the page level when an MQTT message is received 399, you can also set actions at the solution (or project) level. The project-level actions will be triggered when an MQTT message is received on any page of the solution and if no page-level MQTT actions have been defined for that page. The actions at project-level also enable you to use a single set of actions across all pages.

To define project-level actions for MQTT messages, click the **Additional Options** button at the right side of the MQTT Actions setting and then add the actions you want to define.

▼ Broadcast Actions

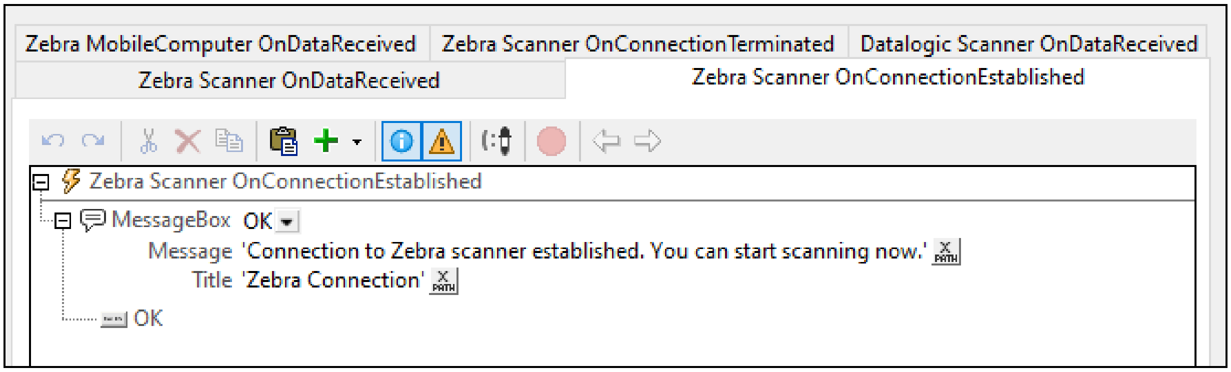
When a broadcast message is received in a solution, actions to perform can be defined in the Broadcast Actions of the project. If no actions have been defined for the **OnBroadcastReceive** 400 page event of the currently active page, then the Broadcast Actions of the project are executed—if defined.

To define the project's Broadcast Actions, click the **Additional Options** button at the right side of the Broadcast Actions setting and then add the actions you want to define.

▼ Barcode Scanner Actions

The Barcode Scanner Actions property defines the actions to perform when events related to barcode scanners are triggered (*see screenshot below*). You can define a sequence of actions for each of the following events, the names of which are self-explanatory:

- Zebra Scanner OnConnectionEstablished
- Zebra Scanner OnDataReceived
- Zebra Scanner OnConnectionTerminated
- Zebra Mobile Computer OnDataReceived
- Datalogic Scanner OnDataReceived


Zebra MobileComputer OnDataReceived
Zebra Scanner OnConnectionTerminated
Datalogic Scanner OnDataReceived
Zebra Scanner OnDataReceived
Zebra Scanner OnConnectionEstablished

Zebra Scanner OnConnectionEstablished
MessageBox OK
Message 'Connection to Zebra scanner established. You can start scanning now.'
Title 'Zebra Connection'
OK

The screenshot above shows the definition of an action to execute when a connection to a Zebra scanner is established (event = Zebra Scanner OnConnectionEstablished).

After the solution connects to a scanner, the data of any barcode that a scanner reads is sent to the solution and stored in the relevant page source tree. The **`OnDataReceived`** event handlers for the different scanners enable you to specify what actions to perform once the barcode data has been received in the page source.

To define the project's barcode scanner actions, click the **Additional Options** button at the right side of the Barcode Scanner Actions setting and then add the actions you want to define. For an overview of how barcode scanners work, see the topic Barcode Scanners[1145].

### Ask User on Exit Workflow

A boolean setting that sets whether the user is asked to confirm (workflow) solution exit. Select `true` or `false` in the combo box. Default value is `true`. If `true`, the text defined as the value of the next property, *Exit Workflow Message*, is displayed before the solution exits. A situation in which the user is asked typically arises if the user presses the **Back**[393] button on the first page of a solution. The user will **not** be asked for an exit-confirmation if the **Submit**[394] button is pressed or if a Cancel Action Execution[913] is processed.

### Exit Workflow Message

The text of the message that is displayed as a prompt to confirm (workflow) solution exit. The message is displayed only if the previous property, *Ask User on Exit Workflow*, is set to `true`. The default message is: *Do you really want to exit this solution?*

### On Switch to Other Solution

While a solution is running, the end-user could switch to another solution. If this happens, the *On Switch to Other Solution* setting determines whether the original solution is suspended (paused and minimized), or canceled. If the solution is suspended, then the solution is paused at that point, and no further solution action is executed: for example, no timers are executed, no geolocations are used. When the solution is resumed, actions defined for the *On Reopen* option of the `OnPageRefresh` 390 event are executed. The setting's options are:

- *Cancel this solution*: The default value. The solution is canceled; any unsaved data will be lost.
- *Suspend this solution*: The solution is paused but is kept open. Its icon will become available in the device's *Running* tab. To switch back to the solution, the end-user clicks the solution's icon in the *Running* tab.

**Note:** To test this property, the solution must be deployed to the server and run from there.

**Note:** Also see the Solution Execution 915 action, which is another way to specify whether a solution is canceled or minimized.

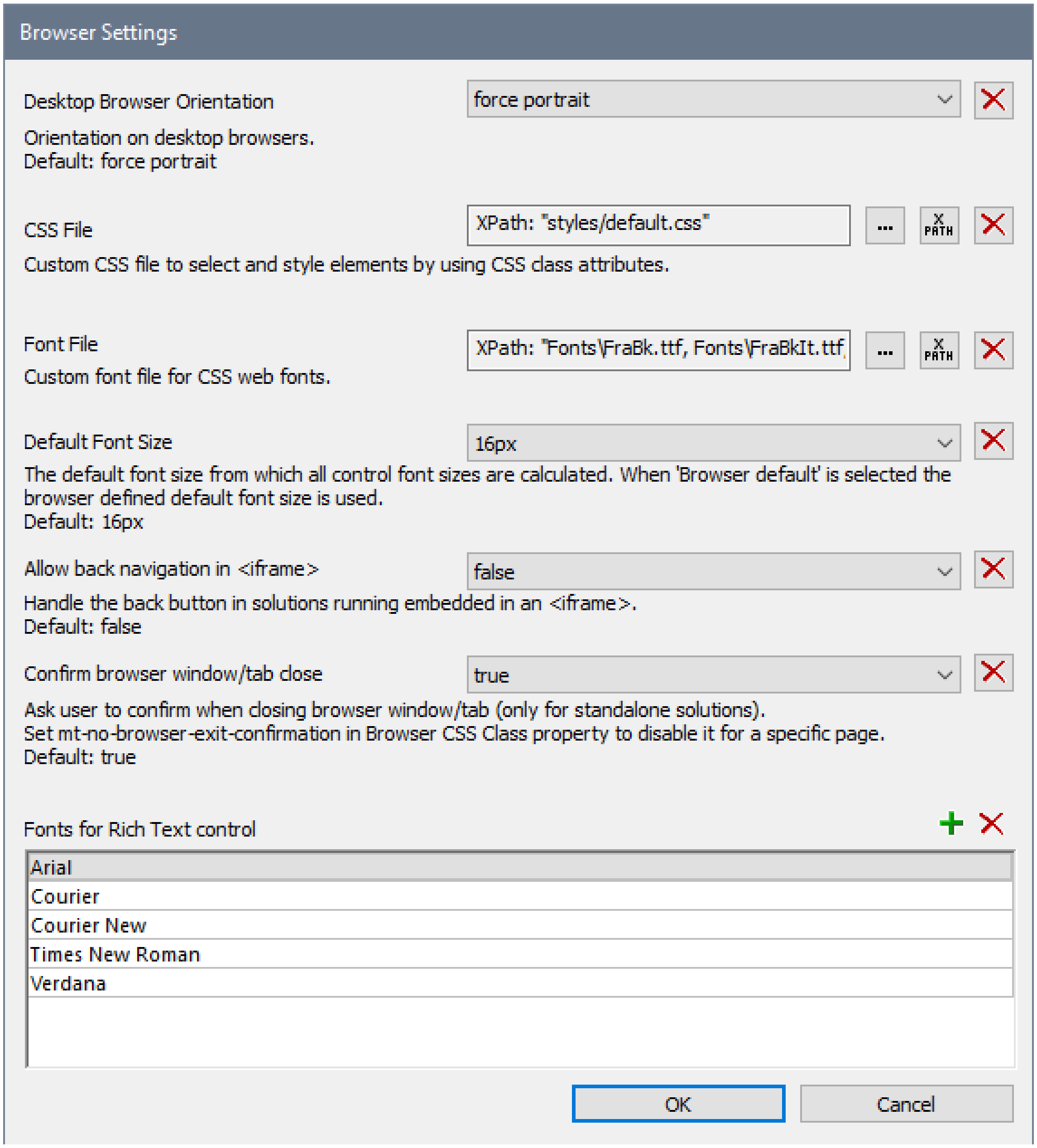
**Note:** Web clients do not support suspended solutions; only the active solution is supported.

▼ Workflow Icon

Clicking the property's **Additional Dialog** button displays a Browse dialog in which you can browse for the PNG image file to use as the icon of the project on client apps. The maximum pixel size of workflow icons is `200x200`. By default, the MobileTogether icon will be used.

▼ Browser Settings

Clicking the *Browser Settings* property's **Additional Dialog** button displays the Browser Settings dialog (*screenshot below*). Here you can define certain settings related to the browser of the mobile device. These settings are described below.



Browser Settings

Desktop Browser Orientation: force portrait

Orientation on desktop browsers.
Default: force portrait

CSS File: XPath: "styles/default.css"

Custom CSS file to select and style elements by using CSS class attributes.

Font File: XPath: "Fonts\FraBk.ttf, Fonts\FraBkIt.ttf,

Custom font file for CSS web fonts.

Default Font Size: 16px

The default font size from which all control font sizes are calculated. When 'Browser default' is selected the browser defined default font size is used.
Default: 16px

Allow back navigation in <iframe>: false

Handle the back button in solutions running embedded in an <iframe>.
Default: false

Confirm browser window/tab close: true

Ask user to confirm when closing browser window/tab (only for standalone solutions).
Set mt-no-browser-exit-confirmation in Browser CSS Class property to disable it for a specific page.
Default: true

Fonts for Rich Text control

Arial
Courier
Courier New
Times New Roman
Verdana

The following settings can be defined:

*Desktop Browser Orientation*
The combo box options enable you to select the orientation of the browser: *Force Portrait* and *Force Landscape*. The default is *Force Portrait*.

*CSS File*
For styling in web clients—that is, in browsers—only, this setting specifies the external CSS file that is read in order to evaluate CSS properties assigned to the class selectors of controls in the design. An external CSS file can be modified at any time in order to change the appearance of design components.

Each design component has a property called `Browser CSS Class`, which defines a CSS class name specific to that control. CSS properties for these class selectors can then be defined in an external CSS file, which is deployed to the server. The CSS file to look up for the class rules is specified in this (*CSS File*) setting. You can select the CSS file via a file path or global resource alias[1334]. You can also use an XPath expression to generate the file path (*see screenshot above*). Note the following points: (i) the CSS rules defined in the external CSS file have a lower priority than the definitions that are made in a control's properties; (ii) the CSS file is not available in simulations for web browsers.

*Font File*
Specifies one or more font files that you want to embed in the design and use in addition to the system fonts. You can either browse for the font file, locate it with a global resource, or generate its filepath with an XPath expression. The following font file types are supported: **`.ttf`**, **`.otf`**, **`.woff`**, **`.woff2`**. Additionally, MobileTogether will correctly generate **`.eot`**, **`.svg`**, and **`.svgz`** fonts; however, these font file types are not supported by all browsers. If you want to embed multiple font files, enter an XPath expression which is a string containing the multiple filepaths separated by commas (*see screenshot above*). A font that is embedded in the design in this way can be referenced via the `font-family` property of CSS. If a font has been embedded that is also available on the local system, then the system font will be used. If you specify the same font in different file types (say, WOFF2 and TTF), then the browser will download the one file type that it supports best, and none of the alternative file types. *For more information about CSS and browser-related font information, see the following MDN web pages: @font-face and font-family.*

*Allow Back Navigation in <iframe>*
This property applies to Embedded Webpage Solutions[1427], which are solutions that are loaded into an **`IFrame`** element of a webpage. If this option is set to **`true`**, the **Back** navigation button of the browser allows the end user to navigate back within the IFrame. The default is **`false`**.

*Default Font Size*
Select the base font size from which all control font sizes are calculated. The *Browser Default* option selects the default size of the browser. The default option is `16px`.

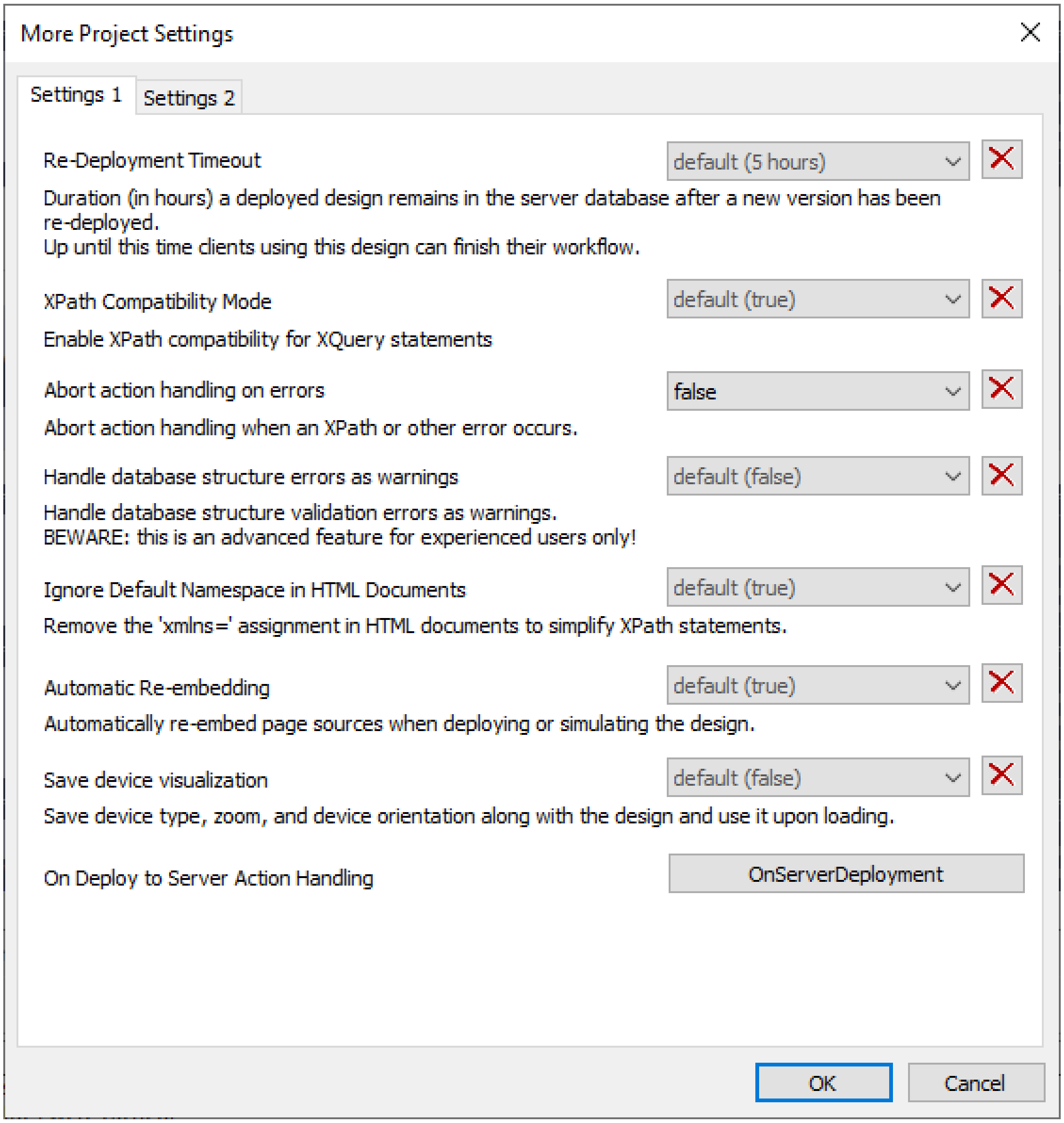
*Confirm Browser Window/Tab Close*
This option enables a message box to be displayed when the end user wants to close the browser window or browser tab that is displaying the solution. The message (i) asks whether the user really wants to leave the page, and (ii) informs the user that if he or she clicks **`Leave`**, then unsaved changes might not be saved. This setting applies to **`all pages`** in the project. If this option is selected, and you want to disable it for one or more pages individually, then you must set the **`Browser CSS Class`**[384] property of each of these pages *(see Page Properties*[384]*)* to a value of `mt-no-browser-exit-confirmation`.

*Fonts for Rich Text Control*
Add the fonts you want to allow the end user to select from. These fonts will be displayed in the dropdown list of the font selection combo box of the Rich Text Control[1233]. If no font is specified in this list, then the font selection combo box will be disabled in the solution.

*Note*
Relative file paths that are set in this dialog are relative to: (i) the solution directory on the server, and (ii) in MobileTogether Designer, to the directory in which the design is located.

▼ More Project Settings

Clicking the *More Project Settings* property's **Additional Dialog** button displays the More Project Settings dialog (*screenshot below*). Note that settings are divided across two tabs.



More Project Settings

Re-Deployment Timeout — default (5 hours)

Duration (in hours) a deployed design remains in the server database after a new version has been re-deployed.
Up until this time clients using this design can finish their workflow.

XPath Compatibility Mode — default (true)

Enable XPath compatibility for XQuery statements

Abort action handling on errors — false

Abort action handling when an XPath or other error occurs.

Handle database structure errors as warnings — default (false)

Handle database structure validation errors as warnings.
BEWARE: this is an advanced feature for experienced users only!

Ignore Default Namespace in HTML Documents — default (true)

Remove the 'xmlns=' assignment in HTML documents to simplify XPath statements.

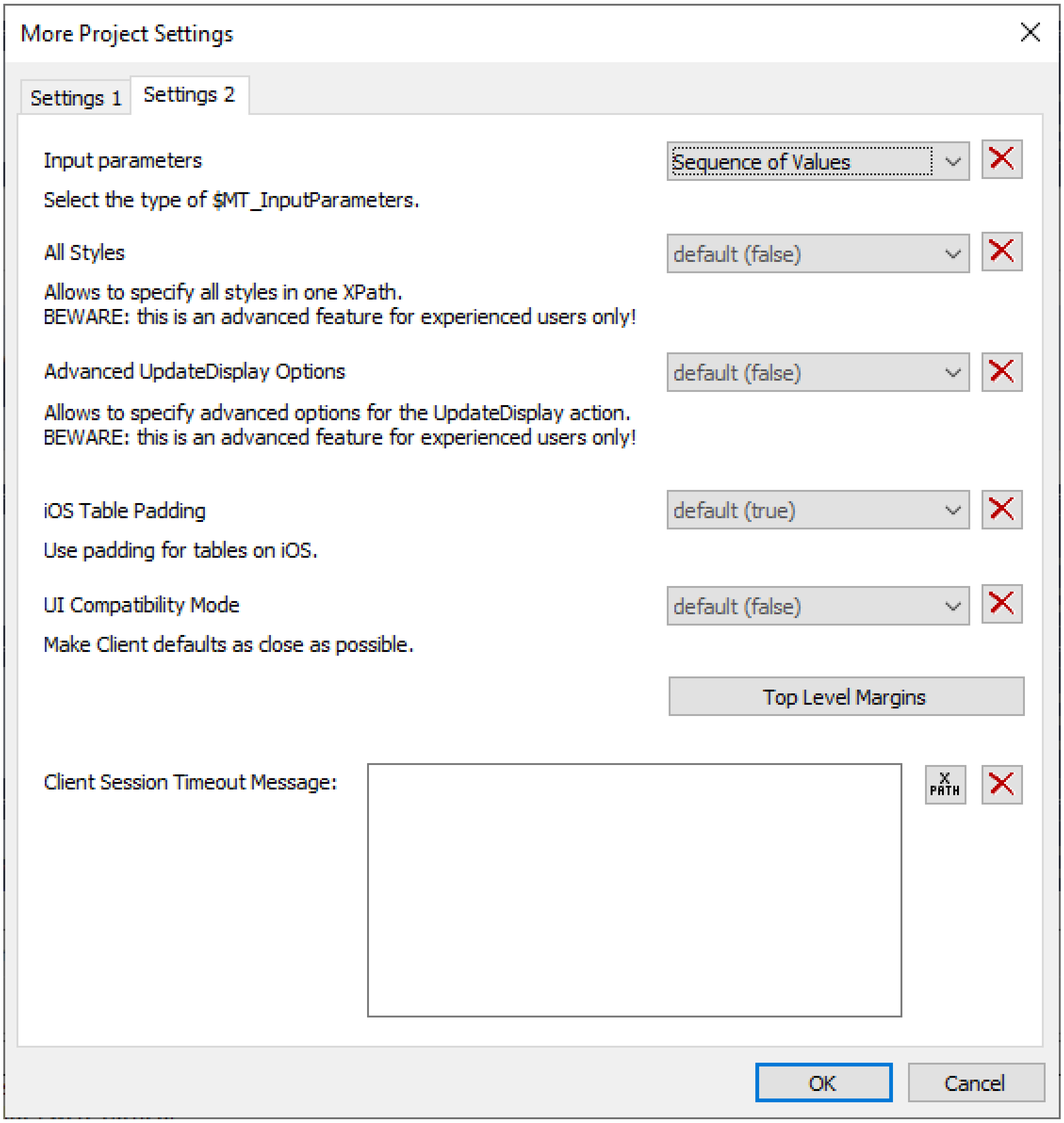
Automatic Re-embedding — default (true)

Automatically re-embed page sources when deploying or simulating the design.

Save device visualization — default (false)

Save device type, zoom, and device orientation along with the design and use it upon loading.

On Deploy to Server Action Handling — OnServerDeployment



More Project Settings

Input parameters: Sequence of Values

Select the type of $MT_InputParameters.

All Styles: default (false)

Allows to specify all styles in one XPath.
BEWARE: this is an advanced feature for experienced users only!

Advanced UpdateDisplay Options: default (false)

Allows to specify advanced options for the UpdateDisplay action.
BEWARE: this is an advanced feature for experienced users only!

iOS Table Padding: default (true)

Use padding for tables on iOS.

UI Compatibility Mode: default (false)

Make Client defaults as close as possible.

Client Session Timeout Message:

The following settings can be defined:

- *Re-deployment Timeout:* The time in hours after a new version of a solution is deployed that the superseded solution is kept on the server. This overlap time enables clients currently using the older solution to complete work. The default is 5 hours.
- *XPath Compatibility Mode:* When set to `true`, XQuery constructs that are invalid in XPath are resolved so that XQuery statements containing these constructs are compatible with XPath and can be used where XPath expressions are allowed. Currently, this concerns XQuery entity and character references, which are allowed in XQuery, but not in XPath. When *XPath compatibility mode* is set to `true`, XQuery entity and character references are read in XPath as text; they are not resolved. The default value for this setting is `true`.

- *Abort Action Handling on Errors:* Aborts the handling of actions when an error occurs. The error could be in an XPath expression or elsewhere in the action handling. However, minor errors such as XPath errors for selecting a style property are ignored and action handling continues. The default value is `true`.

- *Ignore Default Namespace in HTML Documents:* Since only one default namespace is allowed in an XML document, not ignoring the default namespace in HTML documents could create errors in reading XML data sources. The default is `true`: The HTML default namespace is ignored.

- *Automatic Re-embedding:* Embedding[289] refers to the embedding of page sources in the project (design) file. If *Automatic Re-embedding* is enabled (`true`), then page sources are re-embedded when deploying or simulating, ensuring that the latest data source files are embedded and that the data, therefore, is up-to-date. The default is `true`.

- *Save Device Visualization:* If selected, then the device settings[254] (device type, zoom level, and page orientation) will be saved with the design. The design will be re-opened with the last-saved device settings. The default is `false`.

- *On Deploy to Server Action-handling:* On clicking *OnServerDeployment*, an Actions dialog for the `OnServerDeployment` event is opened. You can enter a sequence of actions to be performed when the project is deployed to the server. Only server-related actions can be added to this action tree, and only these actions will be enabled (that is, those actions that are not grayed out). During the action-handling the server is locked, and clients will not be able to connect to it. The values of input parameters for these action are submitted during the deployment step[1574]. During deployment, the parameter values will be passed to the solution's **`$MT_InputParameters`**[1300] variable and can be accessed from there. Note that the data structure of the **`$MT_InputParameters`**[1300] variable is specified in a project setting (*see next setting*).

- *Input Parameters:* Specifies the data structure type of the **`$MT_InputParameters`** variable. The options are: (i) *Named Parameters*, which is a map data structure (for example: `{"name":"Altova", "location":"Boston"}`) and (ii) *Sequence of Values*, which is a sequence data structure (for example: `("Altova", "Boston")`). The default is *Named Parameters*. For more information, see the descriptions of **`MT_InputParameters`**[1300], the previous project setting above, *On Deploy to Server Action-handling*, and the **Deploy to MobileTogether Server**[1574] command.

- *All Styles:* This setting specifies whether the `All Styles` property of a design component or page is available or not. The setting's values are `true` or `false`, with `false` being the default. If set to `true`, then the `All Styles` property will be displayed in the Styles & Properties Pane[274]. The property enables you to set all the styles of the selected component or page conveniently at one location, via an XPath map expression. For information about how to use the `All Styles` property, see its description, for example, on the Button[417] control.

- *Advanced UpdateDisplay Options:* Specifies whether the Update Display action[790] offers advanced options (`true`) or not (`false`). The advanced options enable you to specify which controls of a page are updated. If advanced options are not switched on, then all controls of a page are updated. The default setting is `false`. For details about the Update Display action's options, see Update Display[790].

- *iOS Table Padding:* Specifies whether the standard iOS table padding is applied to tables on iOS devices. The default value of this setting is **`true`**. When the setting is `true`, all tables in the project are given a padding of **`9px`** on right and left, and **`5px`** on top and bottom. If values are also set for any `Padding` property of individual tables, then that `Padding` value (top, right, bottom, or left) is added to the respective iOS table padding value. If *iOS Table Padding* is set to `false`, then the default iOS table padding will not be applied, and only the padding values you set for an individual table will be applied. Note that the *iOS Table Padding* setting applies to iOS tables in the entire project.

- *UI Compatibility Mode:* Some default style properties are handled differently by client devices on different platforms. The *UI Compatibility Mode* setting's default value is **`false`**. Set it to **`true`** if you want the styling defaults of design components to be similar across platforms. As a result, the default of incongruent style values will be modified. For example, iOS table padding is set to `false`, top-level margins are set to `0px` for all devices, and button padding is set to `0px` on Android and `1dp` on other platforms. Note that the *UI Compatibility Mode* setting modifies default styles. You can set custom styles in the usual way 1315. For a list of default style values that are different across platforms, see Style Variance Across Clients 1330.

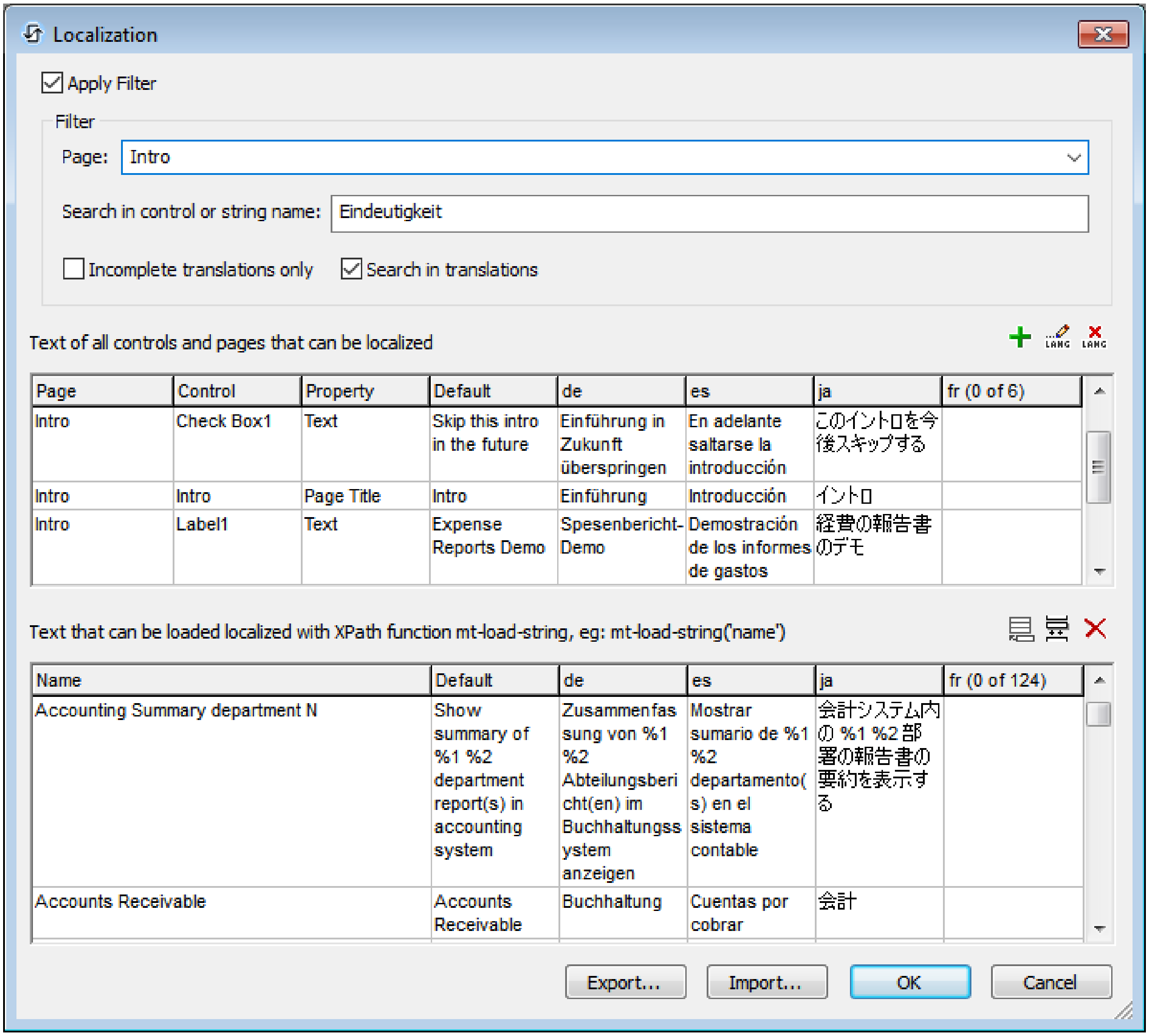
- *Top Level Margins:* Top-level controls are controls that are located directly inside the design—that is, these are any control that is not inside a table. The margin you set in the Top Level Control Margins dialog will override the default device-specific margins. They essentially set a margin for each page of the project, and thus provide you with better control of the layout. For example, Android devices currently set a default margin of 9px (*but see note about Label controls at end of para*); if you want another margin for your project pages, you can use Top Level Margins to adjust the margin. The *Top-Level Control Margin* property *Default for all* sets the specified margin on all four sides. You can also set the top, right, bottom and left margins individually. If a margin setting is left blank, the default device-specific margin is used. (**Note:** Label controls on Android have a bottom margin of **`0px`**. To modify this setting, either change the top level margin setting (this setting) or the bottom margin of the Label control.)

- *Client Session Timeout Message:* You can enter a message to be displayed on the client device when the when the server has timed out. The message can be either entered directly or it can be obtained as the result of evaluating an XPath expression.

# 6.6 Localization

A solution is created in a default language. But the text strings used in the solution can be localized (translated) into multiple languages. When the solution runs in a mobile device, the language of the solution is automatically selected to be the same as that of the mobile device. If the solution has not been localized into the language of the mobile device, the default language of the solution is used (*see screenshot below*).

The localized strings are defined in the Localization dialog (*screenshot below*) by adding a column for each new language and defining the localized strings in this column. For details, see the description of the menu command, **Project | Localization**[1600]. Additionally, named text strings can also be localized and subsequently referenced anywhere in the design by using the `mt-load-string`[1600] extension function in an XPath expression: `mt-load-string(NameOfString')`.



Localization

☑ Apply Filter

Filter

Page: Intro

Search in control or string name: Eindeutigkeit

□ Incomplete translations only ☑ Search in translations

Text of all controls and pages that can be localized

| Page | Control | Property | Default | de | es | ja | fr (0 of 6) |
|---|---|---|---|---|---|---|---|
| Intro | Check Box1 | Text | Skip this intro in the future | Einführung in Zukunft überspringen | En adelante saltarse la introducción | このイントロを今後スキップする | |
| Intro | Intro | Page Title | Intro | Einführung | Introducción | イントロ | |
| Intro | Label1 | Text | Expense Reports Demo | Spesenbericht-Demo | Demostración de los informes de gastos | 経費の報告書のデモ | |

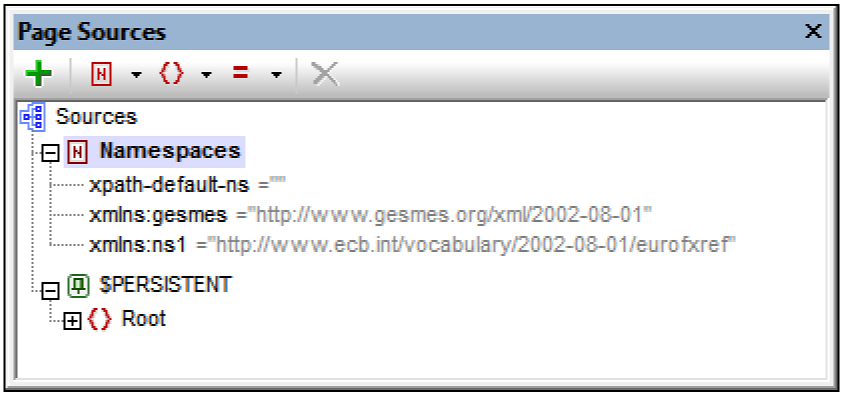
Text that can be loaded localized with XPath function mt-load-string, eg: mt-load-string('name')

| Name | Default | de | es | ja | fr (0 of 124) |
|---|---|---|---|---|---|
| Accounting Summary department N | Show summary of %1 %2 department report(s) in accounting system | Zusammenfassung von %1 %2 Abteilungsbericht(en) im Buchhaltungssystem anzeigen | Mostrar sumario de %1 %2 departamento(s) en el sistema contable | 会計システム内の %1 %2 部署の報告書の要約を表示する | |
| Accounts Receivable | Accounts Receivable | Buchhaltung | Cuentas por cobrar | 会計 | |

Localized solutions can be tested by selecting the simulation language you want via the **Project | Simulation Language** 1606 command and then running a simulation.

## 6.7 Namespaces

Namespaces are important for correctly identifying nodes, and for correctly locating nodes with the use of XPath expressions. The *Namespaces* item in the Page Sources Pane 270 (*screenshot below*) contains all the namespaces that have been declared for the project, irrespective of what page is currently active in Page Design View 253.



Page Sources

Sources
Namespaces
xpath-default-ns =""
xmlns:gesmes ="http://www.gesmes.org/xml/2002-08-01"
xmlns:ns1 ="http://www.ecb.int/vocabulary/2002-08-01/eurofxref"
$PERSISTENT
Root

Namespaces can be declared in two ways:

- *Automatic declaration on data import:* When an external XML file is added as a page source, namespaces in the source are automatically imported into the design and declared for the scope of the entire project. They then appear under the *Namespaces* item in the Page Sources Pane 270 (*see screenshot above*). The namespace prefixes are automatically set to match the original prefixes if such matching creates no ambiguities in the design. Prefixes assigned in the namespace declaration are used in node names, and must be used in XPath expressions that are intended to locate these nodes in the page source.
- *User-defined:* You can also add namespaces by clicking the **Namespace** icon in the toolbar of the Page Sources Pane 270 (*screenshot above*). Being able to add your own namespaces to a project enables you to create nodes that belong to one or more user-declared namespaces. This is useful for disambiguating between nodes that have the same local name.

To delete a namespace, select it and click **Delete** in the pane's toolbar 270.

**Note:** A namespace prefix can be renamed at any time in the design process by double-clicking it in the Page Sources Pane and editing it. All references to the old prefix in XPath expressions throughout the design will be changed to the new prefix.

**Note:** The XPath default namespace (`xpath-default-ns=''`) is used for all XPath/XQuery functions, including extension functions and user-defined functions 1293.

## 6.8 Global Resources

Global Resources in MobileTogether allow you to easily specify different database server names, file locations, or other configuration-specific parameters in a way that lets you easily switch from a development to a production system—and you can do that independently for each mobile solution. Because of this, Global Resources enable you to design and test quickly, and, thus, to save time.

As you develop a mobile solution in MobileTogether Designer, the Global Resources dialog allows you to identify each file, database connection, or directory with a user-defined name that can then be used to reference that resource anywhere in your solution. With Global Resources you can set up different configurations and easily switch your mobile solution modes by selecting a different Global Resources configuration.

Once you deploy a new mobile solution to MobileTogether Server, you can upload this configuration with your solution to control the access of your solution to specific servers even after it has been deployed. Administrators can configure the mobile solution's resource configuration on the fly, so a mobile solution can switch from a testing to a production environment with one click.

For more information about how to work with Global Resources, see the section, Altova Global Resources 1334.

# 6.9 Performance

You can optimize performance of your MobileTogether solutions by being aware of how MobileTogether Server and the MobileTogether Client app work together, how MobileTogether Server can be used to do the most data-intensive processing and store large amounts of data, and how settings to optimize performance can be made in the project design in MobileTogether Designer. This section describes important optimization concepts.

## 6.9.1 Embed XML in Design File

Instead of the solution referencing external XML data sources, any XML data can be embedded directly into the design file. This option is perfect for smaller data sets that are needed on the client side, such as a list of choices for a combo-box or other static data. This data is then transmitted to the client as a part of the overall design file and is always instantly available on the client side every time you run the app. So no additional data transfers between client and server are needed.

To embed a data source in the design file, right-click the data source and select **Embed XML in Design File** 359.

## 6.9.2 Data Querying with XQuery 3.1

Using the powerful XQuery 3.1 language, you can write expressions that significantly reduce the amount of data being transferred between the server and client. Database views, queries, or web service calls to external data sources will yield raw data for a mobile application. This often contains redundancies or may not be the ideal structure for the intended data display in the mobile application. XQuery's powerful FLWOR expressions allow you to easily restructure and regroup the data to ensure the most efficient data transfer from server to client and the most useful presentation in the client application.

This new version of XQuery has several new features, including support for maps, arrays, and data in the JSON format.

You can use the Edit XPath/XQuery Expression Dialog 1244 to create and check XQuery expressions.

## 6.9.3 Data Storage on Servers

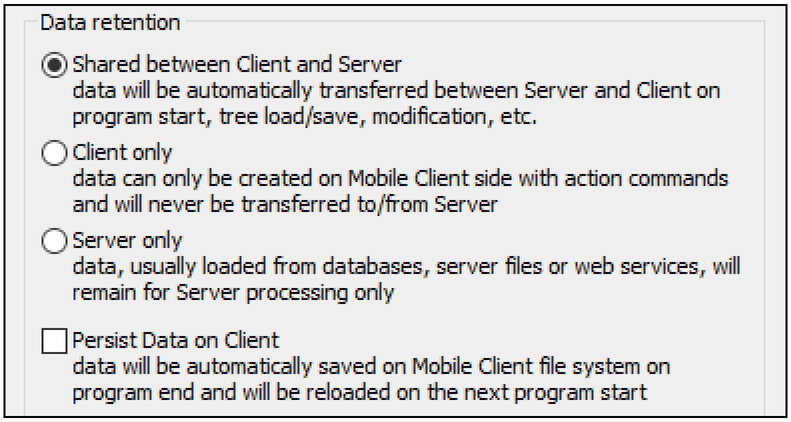
The speed with which data for solutions is processed can be enhanced by storing certain types of data on the server:

- Data that is used to generate charts and graphs, which are images, does not need to be sent to the client; only the image needs to be sent. So the data used to generate the charts and graphs can be kept on the server, and does not need to be transferred.

## Powerful “Keep Data on Server” Setting

To reduce the amount of data transmitted over the mobile data network—which improves the performance of any mobile solution—MobileTogether lets you select exactly which data you want to transmit to the client devices and which data to keep on the server. For example, if a certain data set is only necessary to display a graph, then that data can be kept on the server. The graph image will be rendered by the server and transmitted to the client without the underlying data being transferred over the mobile network. For large data sets this produces a significant performance boost.

This setting (to keep data on the server and not send it to the client) is made for a data source at the time the data source is added. The setting is in the Data Retention pane (*screenshot below*) of the Add Page Source dialog 317. It is also available as a command in the context menus of root nodes 359 in the Page Sources Pane 270.



Data retention

● Shared between Client and Server
data will be automatically transferred between Server and Client on program start, tree load/save, modification, etc.

○ Client only
data can only be created on Mobile Client side with action commands and will never be transferred to/from Server

○ Server only
data, usually loaded from databases, server files or web services, will remain for Server processing only

□ Persist Data on Client
data will be automatically saved on Mobile Client file system on program end and will be reloaded on the next program start

**Note:** For Chart data, do not select the option to keep the chart data on the client only (*Client only*). This will lead to an error.

## Caching

Using settings in MobileTogether Designer and MobileTogether Server, you can specify caching behavior for all data sources. This boosts the speed of MobileTogether greatly because when the server receives a request from the Mobile App, it will already have the data available. There are two main reasons to create caches: (i) If a page data source generates reports slowly (for example, a large database); (ii) If a data source is not modified often. In such cases, execution of a solution would be faster if data is taken from data caches on the server. In order to keep caches up-to-date, the frequency of cache updates can be specified when the cache is created. Once a cache has been defined in MobileTogether Designer, it can be used by the data sources of different designs, providing the underlying data structure is compatible.

As pioneered in Altova MapForce and FlowForce Servers, MobileTogether contains more than just the usual caching parameters such as expiry and refresh time. You can manually determine the amount of time that passes before caching again. Also you can define how many unique combinations of multiple query parameters (either for databases or for web services) should automatically be cached. A client requesting the data will now immediately get it from the cache, whereas the server will retrieve it only if the cache time has elapsed. This is beyond simple caching as MobileTogether actually automatically executes the query to whatever interval the designer specifies. When it is a query with parameters, the designer can specify how many unique combinations of parameters should be cached, and then the server will follow those instructions.

A new cache is defined in MobileTogether Designer for a data source. Right-click a data source in the Page Sources Pane 270, select **Cache Settings**, and specify the properties of the cache.

If a data source is defined as having a cache, the cached data will be used when the solution is run. Caches can be used as soon as the solution has been deployed to the server.

## 6.9.4 Persistent Data Storage on Clients

For user-entered data and data that doesn't change too frequently, you can choose to store data persistently on each client device. This reduces the amount of data being transferred between the server and client, and so increases performance speed. Performance is additionally boosted because the round-trip time between server and client is reduced—even for different sessions of the same user that are hours apart. Persistent data can be defined in the following ways:

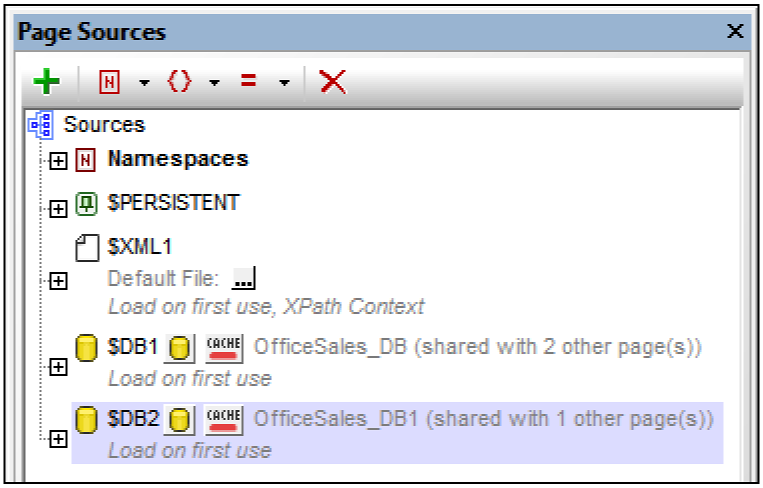
- *Default persistent trees* 349: By default, a `$PERSISTENT` tree is defined for every page in a design. All `$PERSISTENT` tree data is stored on the client. The data can be static or dynamic. If a node in the tree is associated with a control that accepts end-user input, then data in that node of the tree can be edited by the end user.
- *Trees that can be made persistent* 359: In the Page Sources Pane 270, right-click the root node of any tree that is not persistent. In the context menu that appears, select the command **Persist Data on Client**. That tree will be made persistent. The data in the tree will be stored on the client, and will be loaded when the solution starts.
- *Server access on demand* 296: This setting can be defined in the Styles & Properties Pane 274. It specifies that a connection between client device and server is made only when needed. This effectively means that the solution uses persistent data on the client or data that is embedded in the solution. A connection to the server will only be made when specifically required by the design, for example, when the design specifies that data be saved to a database on the server. This approach is very useful for boosting performance when working with databases.

# 7 Page Sources (Data Sources)

A **MobileTogether project** (or design) consists of one or more pages. Each page can have a set of **page sources**, which provide the structure of the structured data that the page will use, as well as, optionally, the actual data that the page will use.

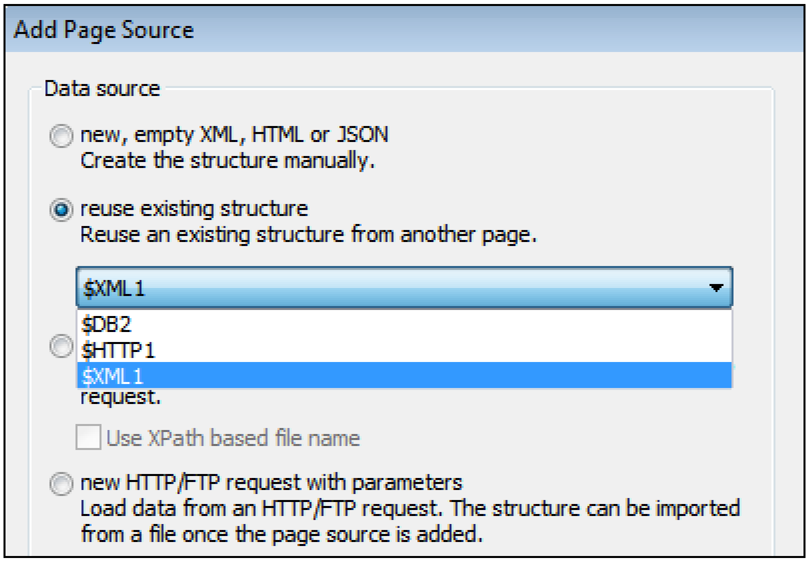
Because page sources are so closely linked with providing data, they are also sometimes called data sources. However, note the following distinction in terminology: The term *data source* is used when referring to the physical entity (like a file or DB) that contains the physical data; the term *page source* is used to cover (i) the general concept of page sources, and (ii) the source of data within the design.

The structure and data of a page's page sources (*see screenshot below left*) are available to controls placed on that page—and that page only. However, a single page source can be reused among multiple pages, enabling the source's data to be accessed on multiple pages of the project. To reuse a page source on another page, create it there as a page source that reuses an existing structure  (*see screenshot below right*).



Page Sources

Sources
- Namespaces
- $PERSISTENT
- $XML1
  Default File: ...
  Load on first use, XPath Context
- $DB1 OfficeSales_DB (shared with 2 other page(s))
  Load on first use
- $DB2 OfficeSales_DB1 (shared with 1 other page(s))
  Load on first use



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

● reuse existing structure
Reuse an existing structure from another page.

$XML1
- $DB2
- $HTTP1
- $XML1

○ request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

## Aspects of page sources

Each page source can be conceptualized as having two aspects:

- *Structure:* The structure of a page source is represented in the form of a tree consisting of element nodes and attribute nodes. The nodes of such a tree can be addressed by using XPath expressions. Data within a node or set of nodes can be: (i) retrieved for display on the page, and (ii) updated (say, by an end user) and placed back inside a node. Both of these operations are possible because the page source is structured and can, as a result, be accessed via XPath.
- *Data:* This refers to the content of the nodes of the page source. The data of all page source nodes is thus available for use in the design. The data in a page source can be obtained from a file. Alternatively, you can assign values to individual page source nodes directly in the Page Sources Pane.This data can be displayed on the page, modified by the end user (on the client device), or processed in specific ways by the design, and then saved back to the page source.

## Number and types and of page sources

Each page can have multiple page sources of one or more kinds. For example, the page represented in the screenshot above left has three page sources: one XML page source and two DB page sources.

The following types of page source can be set up for a page:

- XML Sources 319
- HTML Sources 321
- JSON Sources 323
- HTTP Sources 325
- DB Sources 338
- XQuery Sources 340
- FlowForce Jobs 342

Each type of page source is described in the section Types of Page Sources 317.

## Organization of this section

This section is organized as follows:

- Types of Data Sources 317: describes how to create various types of page sources
- Page Source Properties 345: describes the properties of page sources
- Page Source Trees 347: shows how the structure of a source tree and data in the tree's nodes are used in a page
- Caches 376: describes how the mechanism for caching data on the server works
- Context Menu 359: describes the context menu commands of the Page Sources Pane

# 7.1 Types of Page Sources: Adding

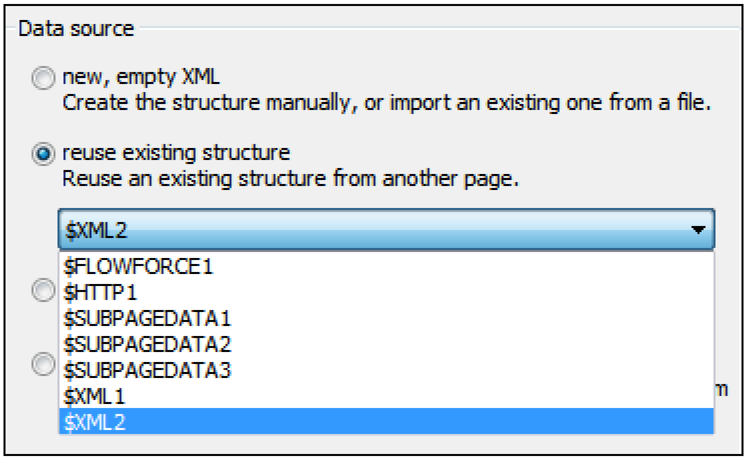
You can add the following types of page sources, each of which is described in the sub-sections of this section

- XML Sources 319
- HTML Sources 321
- HTTP Sources (HTTP/FTP, REST, and SOAP) 325
- DBL Sources 338
- XQuery Sources 340
- FlowForce Jobs 342

After you select the type of page source, you could import the structure of a specific file of that type and, if wanted, data from a specific file. After that you can define the properties of the page source (such as when data from an associated data source file is loaded). You can subsequently, depending on the type of page source, modify the structure, the associated data file, and the properties of a page source, by selecting the appropriate command in the context menu of the page source.

## Reusing existing structures

After a page source has been created for one page of a design (top page or sub page) , it is available for reuse in other pages of the design. If such a page is available, then the *Reuse* option is enabled.



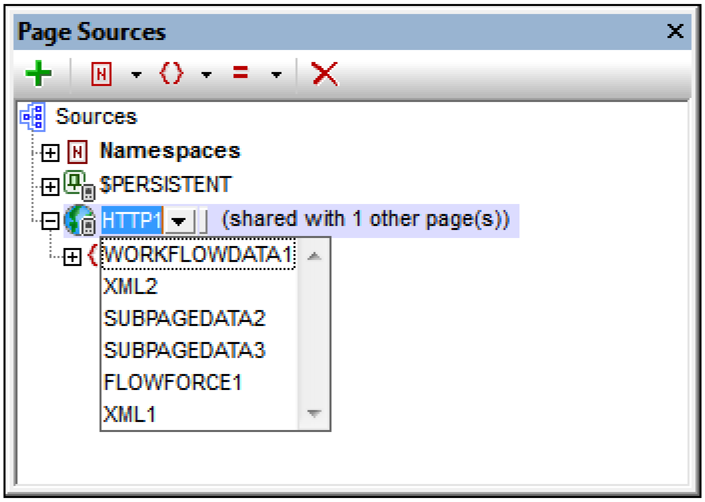
Data source

○ new, empty XML
Create the structure manually, or import an existing one from a file.

● reuse existing structure
Reuse an existing structure from another page.

$XML2

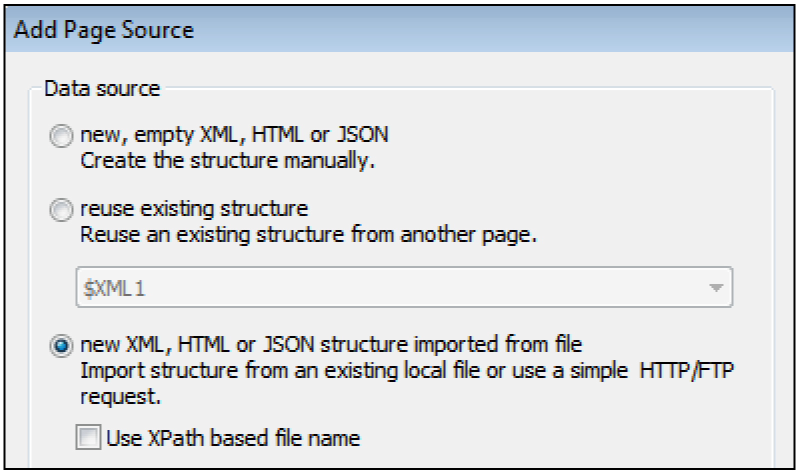
The available page sources are listed by the names of their root nodes in the dropdown list of the options's combo box (*see screenshot above*). Select the page source you want to reuse and click **Finish**. A new root node is created with the same name and structure as the reused page source (*see screenshot below*). The number of pages with which the page source is shared is listed (*see screenshot below*) and the name/s of the sharing pages are displayed when you place your mouse over the root node's name in the tree. You can subsequently change the data structure to that of another page source by selecting another reusable page source in the combo box next to the name of the root node (*see screenshot below*).



How to add data to the tree (including by assigning a default file) is described in the section, Tree Data 354. How to modify the tree structure is described in the section Tree Structure 352.

## Page source structure imported from file

If this option is selected (*see the screenshot below*), clicking **Next** displays the Add Page Source dialog 345, in which you set the usage options of the selected page source. You must specify, in the next screen of the Add Page Source dialog 345, whether the page source is an XML, HTML, or JSON type.


Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

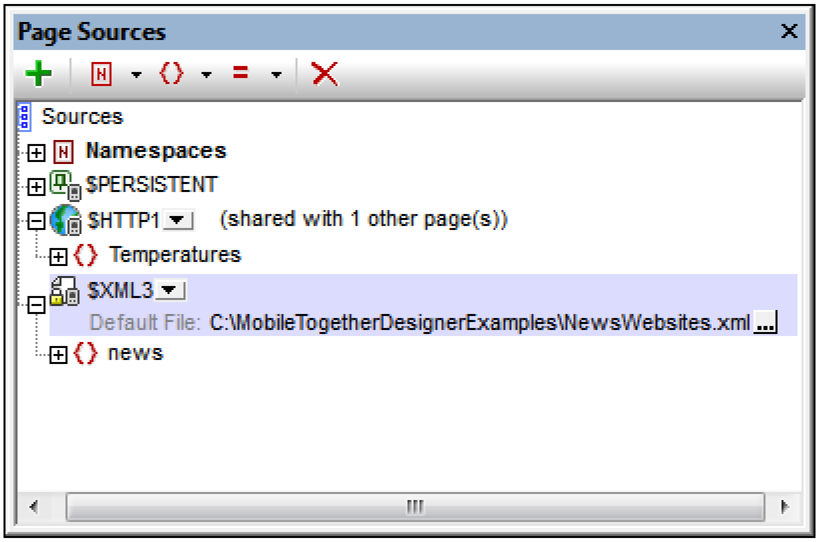
○ reuse existing structure
Reuse an existing structure from another page.

$XML1

● new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

The structure of the XML/HTML/JSON file is imported as the structure of the page source (*see screenshot below*). The structure of an HTML or JSON page source is imported as an XML tree structure. An imported JSON structure will have a root element named `json`. The XML/HTML/JSON (page source) file is also automatically defined as the default file of the page source. This means that data from the file is used as the data of nodes of the new page source. If the file is selected with a URL, you can use the HTTP or FTP protocol to retrieve the file. The path of the default file can also be specified with an XPath expression; this allows the dynamic composition of file paths, for example, paths that are based on node content in other page sources.


Page Sources
Sources
Namespaces
$PERSISTENT
$HTTP1
(shared with 1 other page(s))
Temperatures
$XML3
Default File: C:\MobileTogetherDesignerExamples\NewsWebsites.xml
news

To change the file URL, double-click the URL entry or click the **Additional Dialog** icon at the right-hand side of the entry. If a reusable structure from another project page is available, a combo box next to the name of the root node enables you to select the reusable structure (*see screenshot above*). How to modify the tree structure is described in the section, Tree Structure 352.

Note that HTML retrieval is done using a correcting parser. As a result, if an imported HTML structure has an invalid data object model because of missing elements (according to the HTML 5 specification), then these missing elements are added to the page source tree in the Page Sources Pane. For example:

```
<table>
   <tr/>
   <tr/>
</table>
```

will be corrected to:

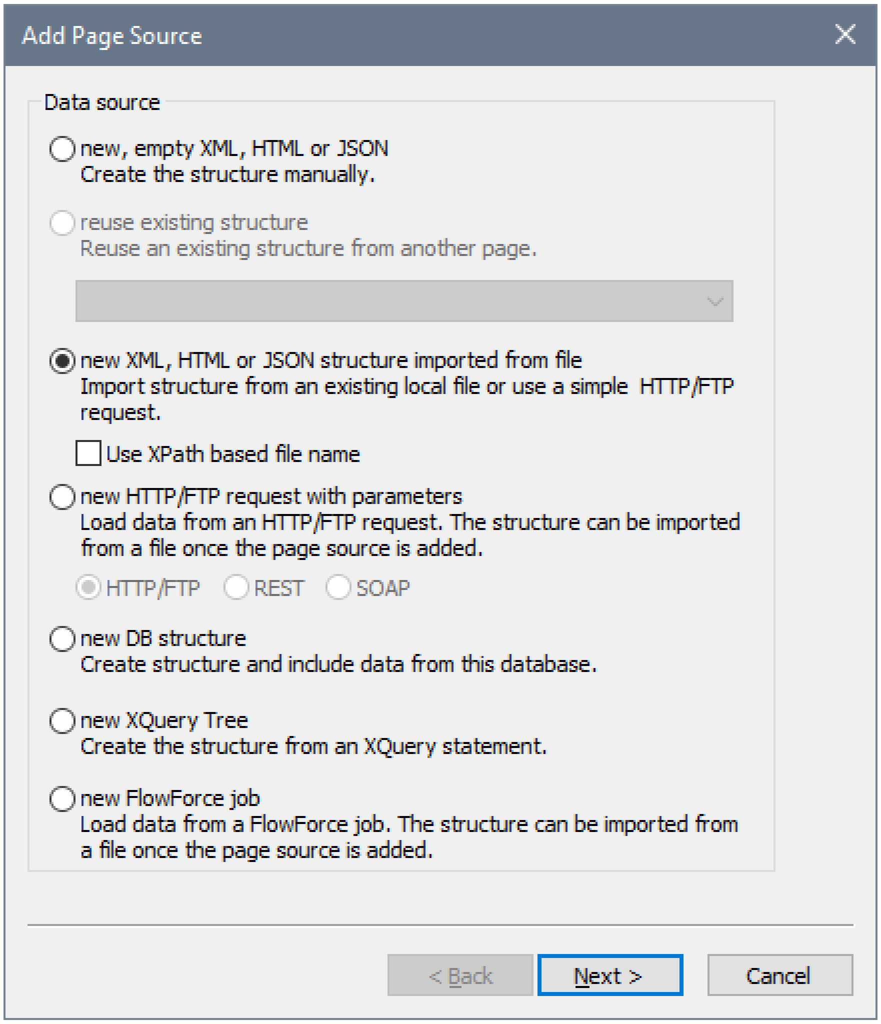
```
<table>
   <tbody>
      <tr/>
      <tr/>
   </tbody>
</table>
```

### 7.1.1 XML Sources

If you want to add a new XML page source, click the **Add Source** toolbar button. This brings up the first screen of the Add Page Source dialog *(screenshot below)*. You now have two options:

- To create the structure manually yourself, select *New, empty XML, HTML or JSON.*
- To import the structure from a file, select *New XML, HTML or JSON structure imported from file.* If the

XPath-based file name option is selected, then the Edit XPath/XQuery Expression dialog[1244] appears and you can build an XPath expression to generate the file URL you need. Otherwise, a dialog box appears in which you can select the file that provides the structure of the page source. You can browse for the file, or use a file URL or global resource.



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

● new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

● HTTP/FTP ○ REST ○ SOAP

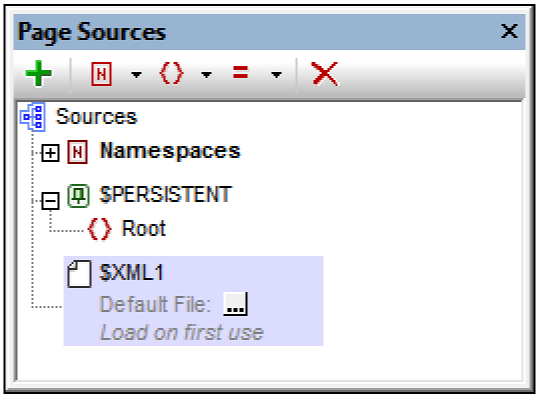
○ new DB structure
Create structure and include data from this database.

○ new XQuery Tree
Create the structure from an XQuery statement.

○ new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

Click **Next** to go to the second screen of the dialog. Here, you (i) specify that the data type of the page source must be XML and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

When you click **Finish**, a root node named **`$XML`** is created for the new page source (*see screenshot below*). You can change the name of the root node if you like by double-clicking in it to edit. If you specified that the structure of the page source must be imported from a file, then, on clicking **Finish**, you will be prompted to select an XML file. The page source **`$XML`** will, in this case, be created with the structure of the selected file.



You can now (i) create or modify the structure of the page source via the toolbar commands, and (ii) add data to nodes of the page source. How to do this is described in the section, Tree Data 354.

**Note:** You can change the data type of the page source (to HTML or JSON) via the root node's context menu command **Data Type** 359.

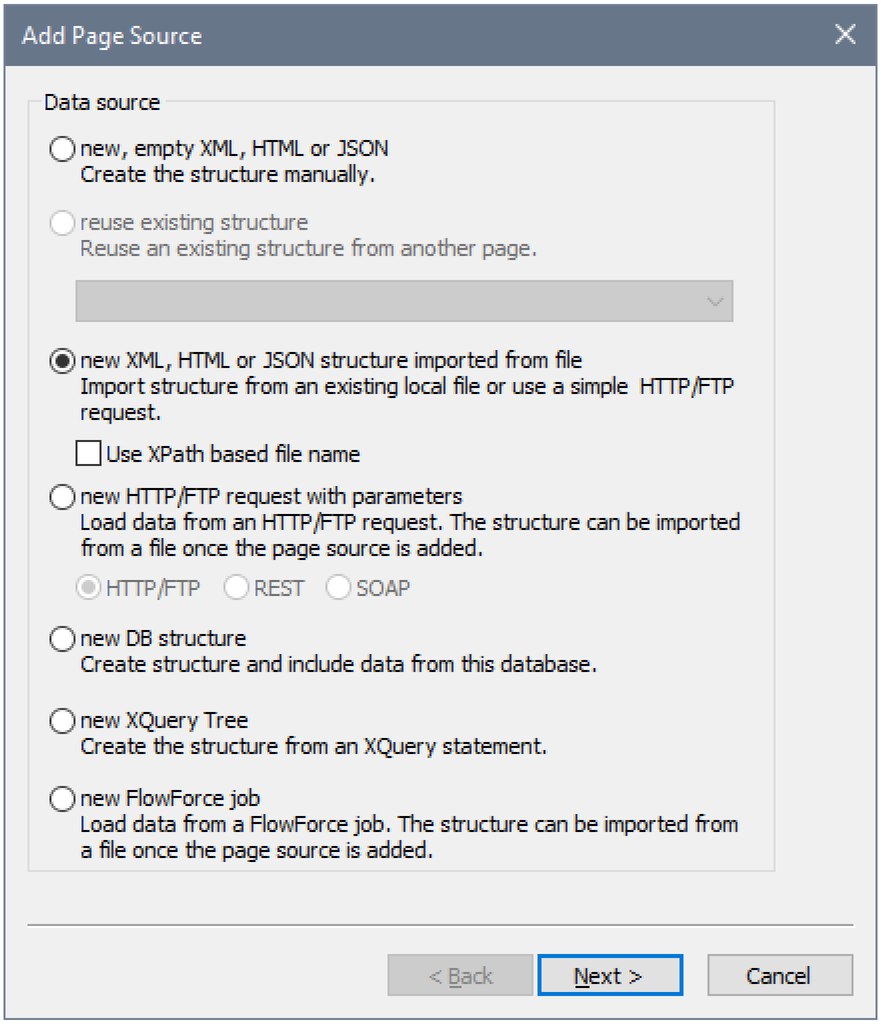
### Working with serialized XML

When dealing with data transfers, you might need to work with serialize XML. MobileTogether's Load/Save String 829 action provides serialization functionality geared to use in its page sources. Serialization of JSON data is discussed in JSON Sources 323.

## 7.1.2 HTML Sources

If you want to add a new HTML page source, click the **Add Source** toolbar button. This brings up the first screen of the Add Page Source dialog *(screenshot below)*. You now have two options:

- To create the structure manually yourself, select *New, empty XML, HTML or JSON.*
- To import the structure from a file, select *New XML, HTML or JSON structure imported from file.* If the XPath-based file name option is selected, then the Edit XPath/XQuery Expression dialog 1244 appears and you can build an XPath expression to generate the file URL you need. Otherwise, a dialog box appears in which you can select the file that provides the structure of the page source. You can browse for the file, or use a file URL or global resource.



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

● new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

● HTTP/FTP ○ REST ○ SOAP

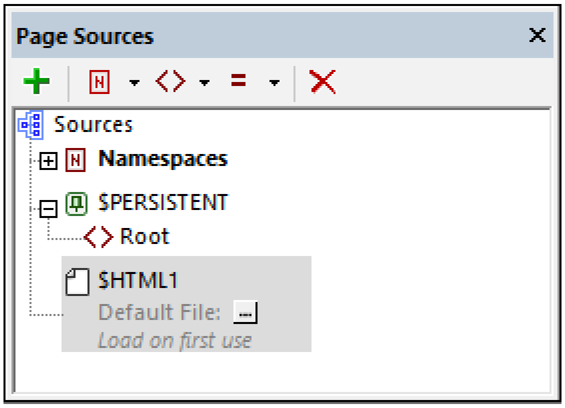
○ new DB structure
Create structure and include data from this database.

○ new XQuery Tree
Create the structure from an XQuery statement.

○ new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

Click **Next** to go to the second screen of the dialog. Here, you (i) specify that the data type of the page source must be HTML and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

When you click **Finish**, a root node named **`$HTML`** is created for the new page source (*see screenshot below*). You can change the name of the root node if you like by double-clicking in it to edit. If you specified that the structure of the page source must be imported from a file, then, on clicking **Finish**, you will be prompted to select an HTML file. The page source **`$HTML`** will, in this case, be created with the structure of the selected file.



You can now (i) create or modify the structure of the page source via the toolbar commands, and (ii) add data to nodes of the page source. How to do this is described in the section, Tree Data[354].

Note that HTML retrieval is done using a correcting parser. As a result, if an imported HTML structure has an invalid data object model because of missing elements (according to the HTML 5 specification), then these missing elements are added to the page source tree in the Page Sources Pane. For example:

```
<table>
   <tr/>
   <tr/>
</table>
```

will be corrected to:

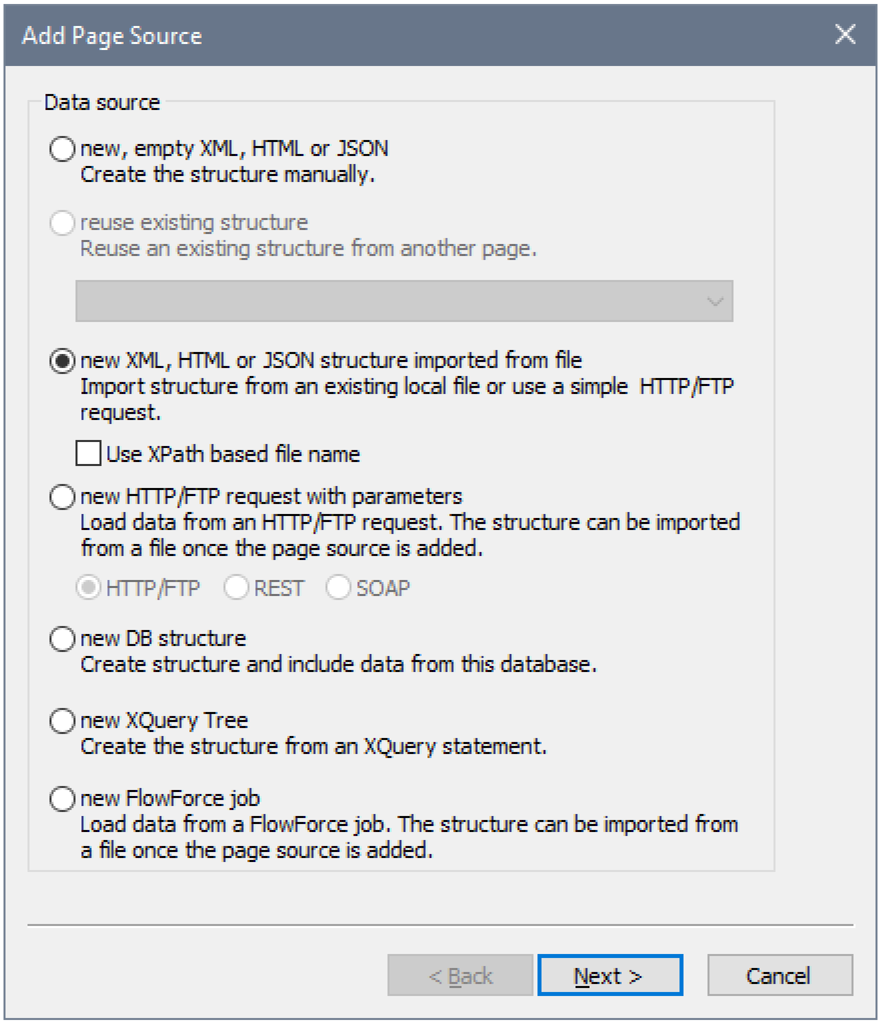
```
<table>
   <tbody>
      <tr/>
      <tr/>
   </tbody>
</table>
```

**Note:** You can change the data type of the page source (to XML or JSON) via the root node's context menu command **Data Type**[359].

## 7.1.3 JSON Sources

If you want to add a new JSON page source, click the **Add Source** toolbar button. This brings up the first screen of the Add Page Source dialog *(screenshot below)*. You now have two options:

- To create the structure manually yourself, select *New, empty XML, HTML or JSON.*
- To import the structure from a file, select *New XML, HTML or JSON structure imported from file.* If the XPath-based file name option is selected, then the Edit XPath/XQuery Expression dialog[1244] appears and you can build an XPath expression to generate the file URL you need. Otherwise, a dialog box appears in which you can select the file that provides the structure of the page source. You can browse for the file, or use a file URL or global resource.



○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

● new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

● HTTP/FTP ○ REST ○ SOAP

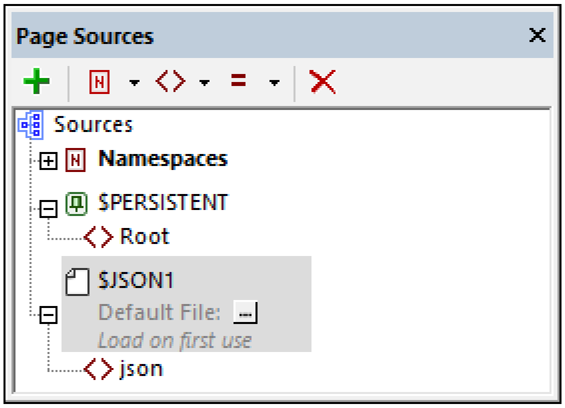
○ new DB structure
Create structure and include data from this database.

○ new XQuery Tree
Create the structure from an XQuery statement.

○ new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

Click **Next** to go to the second screen of the dialog. Here, you (i) specify that the data type of the page source must be JSON and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

When you click **Finish**, a root node named **`$JSON`** is created for the new page source (*see screenshot below*). You can change the name of the root node if you like by double-clicking in it to edit. If you specified that the structure of the page source must be imported from a file, then, on clicking **Finish**, you will be prompted to select a JSON file. The page source **`$JSON`** will, in this case, be created with the structure of the selected file.



You can now (i) create or modify the structure of the page source via the toolbar commands, and (ii) add data to nodes of the page source. How to do this is described in the section, Tree Data[354].

**Note:** You can change the data type of the page source (to XML or HTML) via the root node's context menu command **Data Type**[359].

### Functions to convert JSON to/from string

MobileTogether Designer has two application-specific XPath extension functions to (i) save JSON data to a string, and (ii) load a JSON structure from a string. This is useful, if during solution execution, you want to import or export JSON data, or if you want to obtain the JSON data in a structured form that you can then address with XPath expressions. The two functions are:

- **`mt-save-json-to-string()`**: The function takes a JSON node as its single argument, and returns the contents of the node as a string that is serialized in JSON notation. The submitted JSON node could be an entire JSON document, or a part of a JSON document. *See MobileTogether Extension Functions[1262] for details.*
- **`mt-load-json-from-string()`**: The function takes a string that is a serialized JSON structure as its argument, converts it to XML, wraps it in an element named **`json`**, and returns the whole as a document node. Since a document node is returned, you can append an XPath expression to the function if you want to load a part of the JSON structure. *See MobileTogether Extension Functions[1262] for details.*

Also see the description of the Load/Save String[829] action for more serialization options.

## 7.1.4 HTTP Sources

This section describes the settings needed to make HTTP/FTP, REST, and SOAP requests. These requests are made either to load data from external sources or to save data to external sources. The requests are made in the following situations:

- When adding page sources[317]: In this situation, requests are typically made to load data from external sources

- When defining actions related to page sources [799]: Actions can be specified for page events and control events, and the requests in these actions can be used to either load from or save to external data sources

This section describes the respective dialogs for:

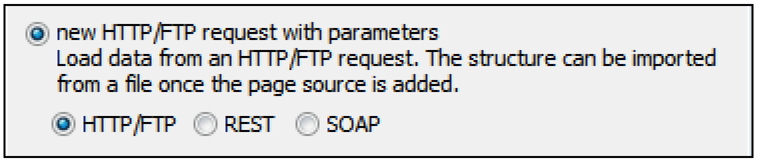
- HTTP/FTP request settings [326]
- REST request settings [328]
- SOAP request settings [337]

### Creating page source structures

The requests that are defined in these settings dialogs are saved in the design, and will be executed at runtime. The page sources will be created but will not contain a tree structure. In order to create a structure, you can import the structure from an XML file or create the structure manually. For example, you can save a SOAP response as an XML file and then import the XML file to generate the tree structure of the page source. See the section Page Source Trees [347] for more information.

### Adding page sources

A page source containing data can be added via an HTTP or FTP request and the structure added subsequently. After you select this option you can specify whether the page source will be obtained using HTTP/FTP, REST, or SOAP (*see screenshot below*). If you select HTTP/FTP or REST, you must specify, in the next screen of the Add Page Source dialog [345], whether the page source is an XML, HTML, or JSON file. You can subsequently change your selection in the respective Settings dialog (HTTP/FTP [326] or REST [328]). (If you select SOAP, the page source must be parsed as XML; this option is automatically set and cannot be changed.)



● new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

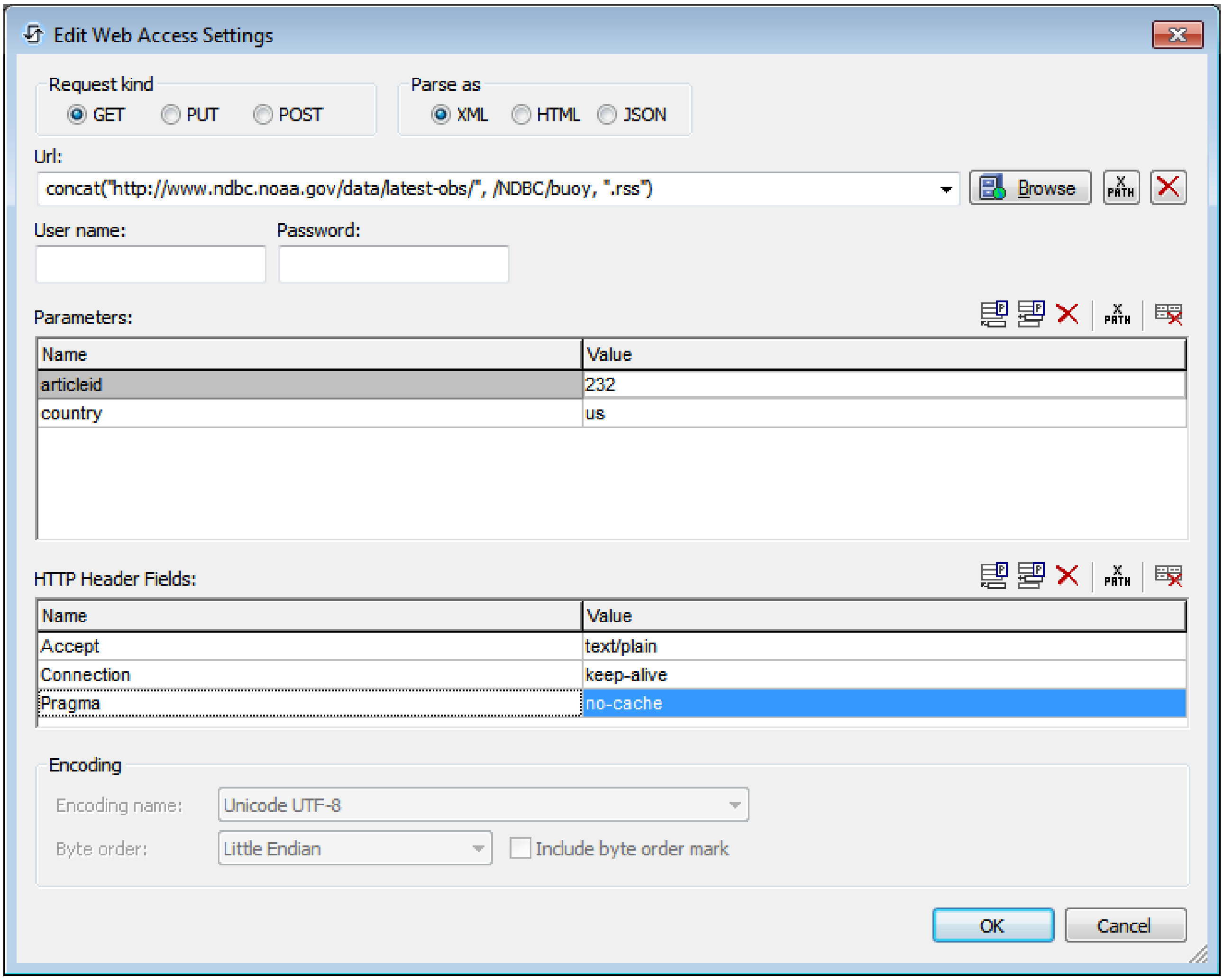
● HTTP/FTP ○ REST ○ SOAP

On clicking the **Finish** button of the Add Page Source dialog, the Edit Web Access Settings dialog [326] (for HTTP/FTP requests), RESTful API Request dialog [328] (for REST requests), or WSDL File Selection dialog [337] (for SOAP requests) is displayed. See the section HTTP/FTP, REST, and SOAP Requests [325] for a description of how to specify the settings of these requests.

If the request is carried out successfully, the page source is added (as a root node [349]) and data from the page source is loaded. The tree structure, however, is not created. It can be imported and/or created manually. How to create the tree structure is described in the section Tree Structure [352].

## 7.1.4.1 Via HTTP/FTP Requests

The settings for HTTP/FTP requests are defined in the Edit Web Access Settings dialog (*screenshot below*). Enter the request kind, the URL of the target resource, the data format of the target resource (XML, HTML, or JSON), user authentication information, and, optionally, query parameters and headers. In the screenshot below, for example, the `GET` request is composed using an XPath expression: It targets a `.rss` page on the `http://www.ndbc.noaa.gov` website. The name of the RSS page is taken from the `/NDBC/buoy` node, and the target page will be parsed as XML. Query parameters and headers can be added to the request. The `charset` header, however, is automatically generated by MobileTogether Designer and will not be overwritten by a `charset` header that you enter in this dialog.



Request kind: ● GET ○ PUT ○ POST

Parse as: ● XML ○ HTML ○ JSON

Url: concat("http://www.ndbc.noaa.gov/data/latest-obs/", /NDBC/buoy, ".rss")

User name:

Password:

Parameters:

| Name | Value |
|---|---|
| articleid | 232 |
| country | us |

HTTP Header Fields:

| Name | Value |
|---|---|
| Accept | text/plain |
| Connection | keep-alive |
| Pragma | no-cache |

Encoding

Encoding name: Unicode UTF-8

Byte order: Little Endian □ Include byte order mark

On clicking **OK**, the request is carried out.

Note that HTML retrieval is done using a correcting parser. As a result, if an imported HTML structure has an invalid data object model because of missing elements (according to the HTML 5 specification), then these missing elements are added to the page source tree in the Page Sources Pane. For example:

```
<table>
   <tr/>
   <tr/>
</table>
```

will be corrected to:

```
<table>
   <tbody>
```

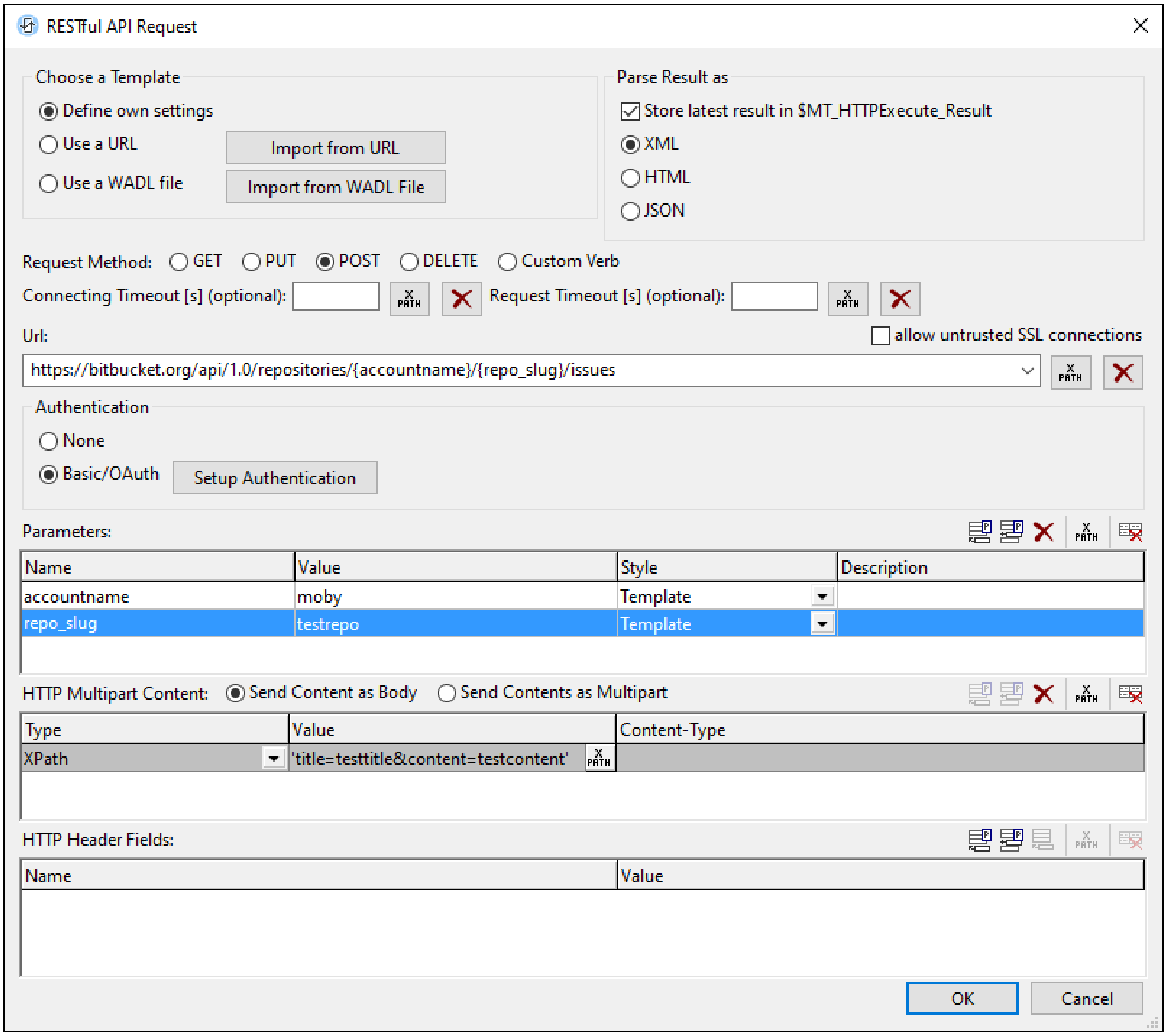
```
      <tr/>
      <tr/>
   </tbody>
</table>
```

### 7.1.4.2 Via REST Requests

The settings for REST requests are defined in the RESTful API Request dialog (*screenshot below*). This dialog is accessed in two situations:

- When defining a page source that uses a REST request to connect to a data source
- Via the Execute REST Request 837 action; in this case the response to the request can be stored in a `$MT_HTTPExecute_Result` variable.

You can choose to (i) define your own settings, (ii) import a URL, or (iii) use a WADL file. If you choose to define your own settings, you can specify your own definitions for the individual settings. If you import a URL or use a WADL file, some settings will be defined in the URL or WADL file, and so cannot be modified by you.



RESTful API Request

Choose a Template

- ● Define own settings
- ○ Use a URL — Import from URL
- ○ Use a WADL file — Import from WADL File

Parse Result as

- ☑ Store latest result in $MT_HTTPExecute_Result
- ● XML
- ○ HTML
- ○ JSON

Request Method: ○ GET ○ PUT ● POST ○ DELETE ○ Custom Verb

Connecting Timeout [s] (optional): Request Timeout [s] (optional):

Url: □ allow untrusted SSL connections

https://bitbucket.org/api/1.0/repositories/{accountname}/{repo_slug}/issues

Authentication

- ○ None
- ● Basic/OAuth — Setup Authentication

Parameters:

| Name | Value | Style | Description |
|---|---|---|---|
| accountname | moby | Template | |
| repo_slug | testrepo | Template | |

HTTP Multipart Content: ● Send Content as Body ○ Send Contents as Multipart

| Type | Value | Content-Type |
|---|---|---|
| XPath | 'title=testtitle&content=testcontent' | |

HTTP Header Fields:

| Name | Value |
|---|---|

OK Cancel

The various settings are described below:

- Templates and result parsing (329), including `$MT_HTTPExecute_Result`
- Request method (332)
- URL (332)
- Authentication (332)
- Parameters (334)
- HTTP content (336)
- HTTP header fields (336)

## Templates and result parsing

With templates are meant the three frameworks within which settings can be defined (your own settings, URL, or WADL; *see screenshot above*). You can switch between templates at any time by selecting the appropriate radio button. If the URL or WADL template option is already selected and you wish to use a different URL or

WADL file, click, respectively, **Import from URL** or **Import from WADL**. Selecting the URL or WADL option (or their respective **Import** button) opens a dialog in which you can specify the URL or WADL file to use.

*Define own settings*

If you define your own settings, you can enter the request to a REST server as a URL, specify the data format of the target resource (XML, HTML, or JSON), then enter user authentication information, and, where appropriate, query parameters, and HTTP content and headers. See the descriptions below of each of these settings.

*Use a URL*

If a URL is long or complex, then it is better to import the URL in the *Use a URL* template rather than enter it in the *Define Own Settings* template. For example, a URL can contain a number of parameters, as in the example below (which is a Google query that contains five parameters):

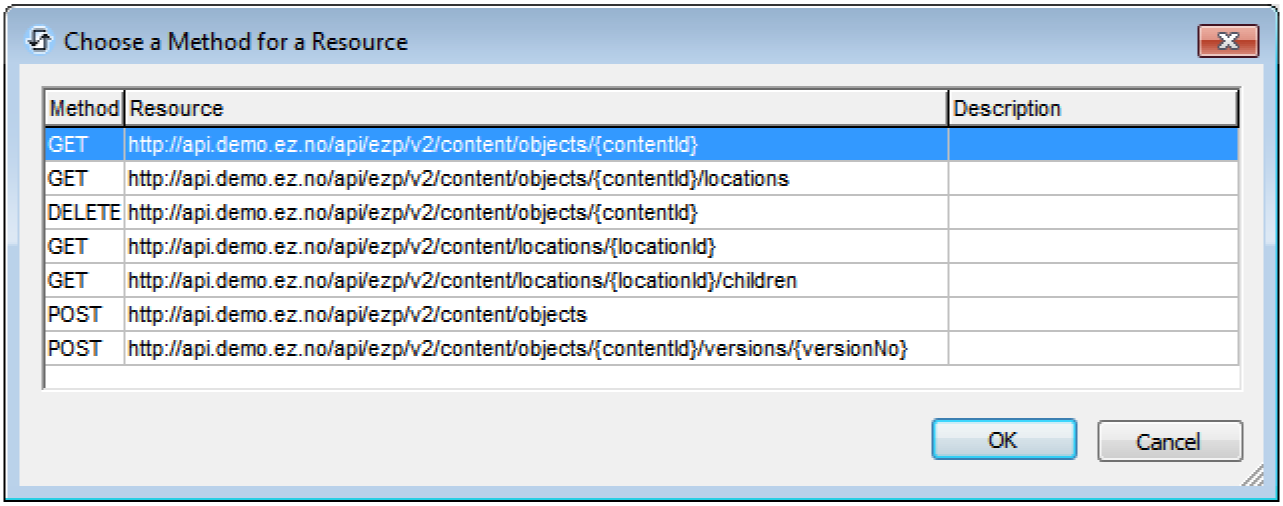
```
https://www.google.at/search?q=REST+WADL&ie=utf-8&oe=utf-
8&gws_rd=cr&ei=89cDVrDHMIP0Up_5vcAB
```

If you import this URL, it is entered in the template's URL field, and the parameters are entered automatically in the template's Parameters table. You can select the data format of the target resource (XML, HTML, or JSON) and the *Request Method* (`GET`, `PUT`, `POST`, or `DELETE`). You can then enter values for the parameters, but you cannot delete a parameter or change its type. If you wish to change the URL, click **Import from URL**. For a description of parameters, see the Parameters section 334 below.

*Use a WADL file*

A WADL file is an XML document that defines the resources provided by a web service and the relationships between the resources. Resources are defined by `resource` elements. Each resource contains: (i) `param` elements to describe the inputs of a resource, and (ii) `method` elements to describe the request and response of a resource. The `request` element specifies how to represent the input, what types are required and any specific HTTP headers that are required. The `response` element describes the service's response, as well as error information.

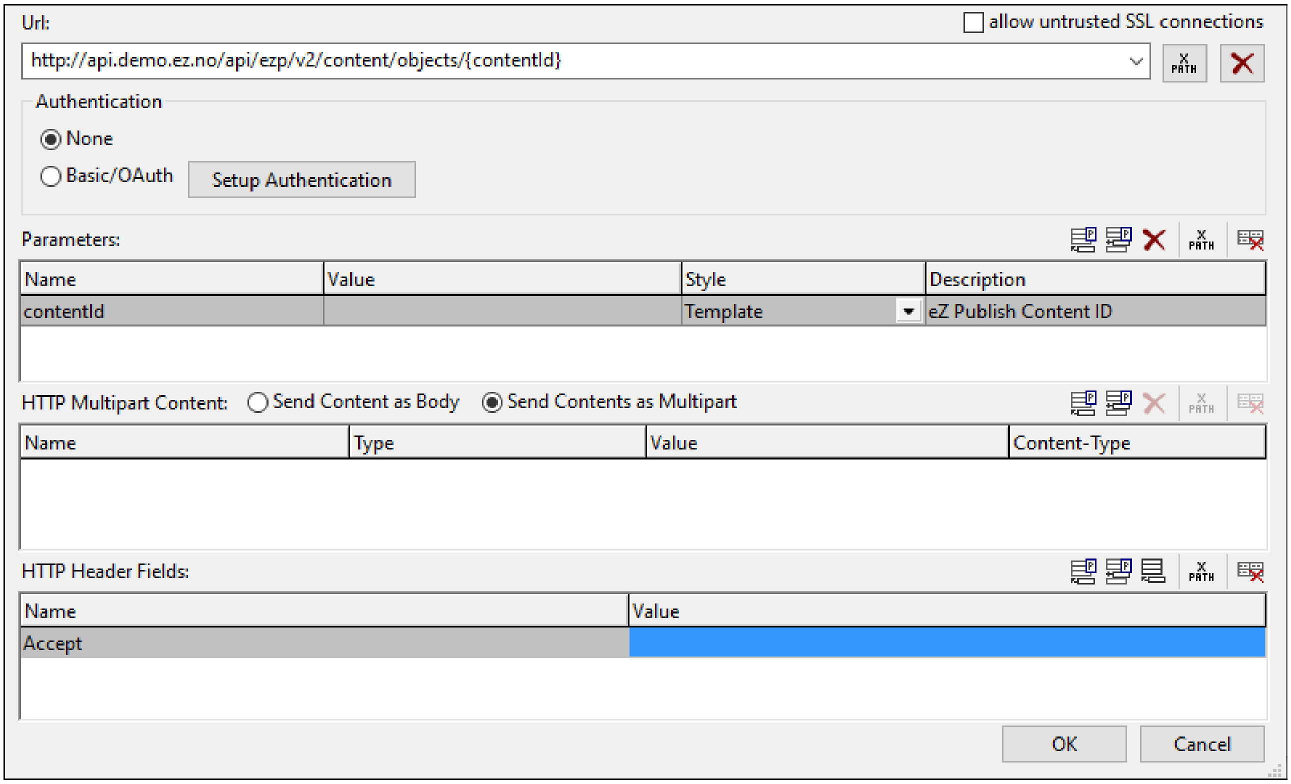
When you select the *WADL* option you are prompted for the WADL file you want to use. After clicking **OK**, the Choose Method dialog appears (*screenshot below*). This dialog displays the methods defined in the WADL file.



Choose a Method for a Resource

| Method | Resource | Description |
|---|---|---|
| GET | http://api.demo.ez.no/api/ezp/v2/content/objects/{contentId} | |
| GET | http://api.demo.ez.no/api/ezp/v2/content/objects/{contentId}/locations | |
| DELETE | http://api.demo.ez.no/api/ezp/v2/content/objects/{contentId} | |
| GET | http://api.demo.ez.no/api/ezp/v2/content/locations/{locationId} | |
| GET | http://api.demo.ez.no/api/ezp/v2/content/locations/{locationId}/children | |
| POST | http://api.demo.ez.no/api/ezp/v2/content/objects | |
| POST | http://api.demo.ez.no/api/ezp/v2/content/objects/{contentId}/versions/{versionNo} | |

Select the method you wish to use, and click **OK**. The URL of the resource is entered in the URL field of the template, and the parameters, and HTTP content and headers defined in the WADL file for that resource are entered in the respective tables of the template (*see screenshot below*). In the template, you can enter the values of parameters and HTTP content and headers, but parameter names cannot be edited and parameters cannot be deleted.



Url: □ allow untrusted SSL connections

http://api.demo.ez.no/api/ezp/v2/content/objects/{contentId}

Authentication

● None

○ Basic/OAuth

Parameters:

| Name | Value | Style | Description |
|---|---|---|---|
| contentId | | Template | eZ Publish Content ID |

HTTP Multipart Content: ○ Send Content as Body ● Send Contents as Multipart

| Name | Type | Value | Content-Type |
|---|---|---|---|

HTTP Header Fields:

| Name | Value |
|---|---|
| Accept | |

*Parse Result as*
The result is what is returned by the web service in response to the request. In the *Own Settings* and *URL* templates, you must specify how this result is to be parsed (as XML, HTML, or JSON) so that MobileTogether can process the result correctly. In the WADL template, the information about the result format is taken from the definitions in the WADL file and automatically selected; as a result these radio buttons are disabled in this template.

*$MT_HTTPExecute_Result*
You can choose whether the result should be stored in the `$MT_HTTPExecute_Result` or not. If stored, you can use the result, via this variable, at other locations in the design. Note that this option is not displayed in the RESTful API Request dialog when the page source is being defined for the first time. Note that this option is displayed only when the dialog is opened via the Execute REST Request 837 action.

## Request method

In the *Own Settings* and *URL* templates, you must specify the Request method (`GET`, `PUT`, `POST`, `DELETE`, or some other verb, which we call a custom verb). A custom verb would be one that is used by the server from which you are requesting the data source and the one that you want to use for the request. To use a custom verb, select its option and enter the custom verb in the text field that appears. You can, instead of directly entering the custom verb, use an XPath expression to generate the custom verb.

In the WADL template, the request method is determined by the selection you make in the Choose a Method for a Resource dialog (*see above for screenshot*), and is automatically selected in the template; as a result these radio buttons are disabled in this template.

You can set the following timeouts:

- A connection timeout, which is the time until a TCP connection is established.
- A request timeout, which is the time to complete the request.

Both values can be entered either directly or via an XPath expression and must be given in seconds, with decimals to be used for fractions of a second. The smallest fraction that will be accepted is the millisecond.

## URL

The URL field can be edited only in the *Define Own Settings* template. In this template, you can enter the URL directly, or as an XPath expression. Use the **Reset** button to clear the entry in the URL field.

*Allow untrusted SSL connections*
A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

## Authentication

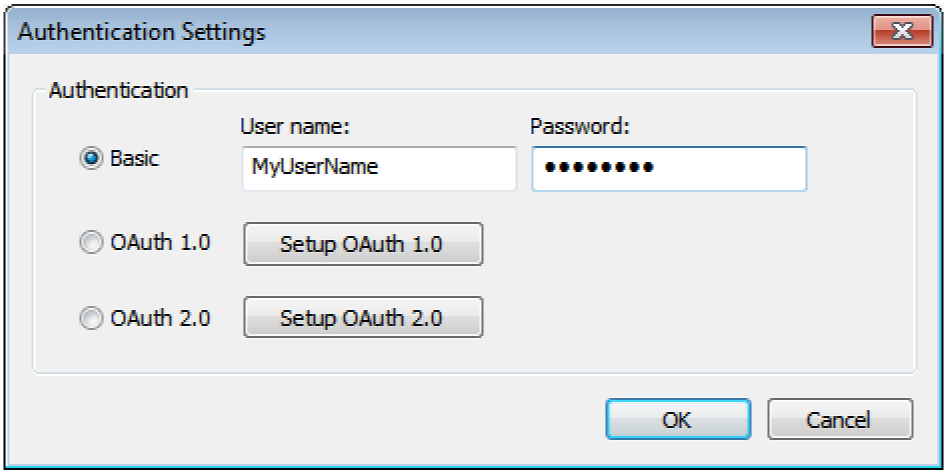
You can supply authentication information if this is required for accessing the server. If no authentication is required, select *None* in the RESTful API Request dialog (*see screenshot above*).

Authentication information can be supplied in the following ways:

- Basic: A user name and password is supplied (*see screenshot below*)

- OAuth 1.0
- OAuth 2.0

If you wish to set up authentication information, click **Setup Authentication** in the RESTful API Request dialog (*see screenshot above*). This displays the Authentication Settings dialog (*screenshot below*), in which you can select the type of authentication the server requires and then supply the authentication details.



● Basic — User name: MyUserName — Password: •••••••••

○ OAuth 1.0

○ OAuth 2.0

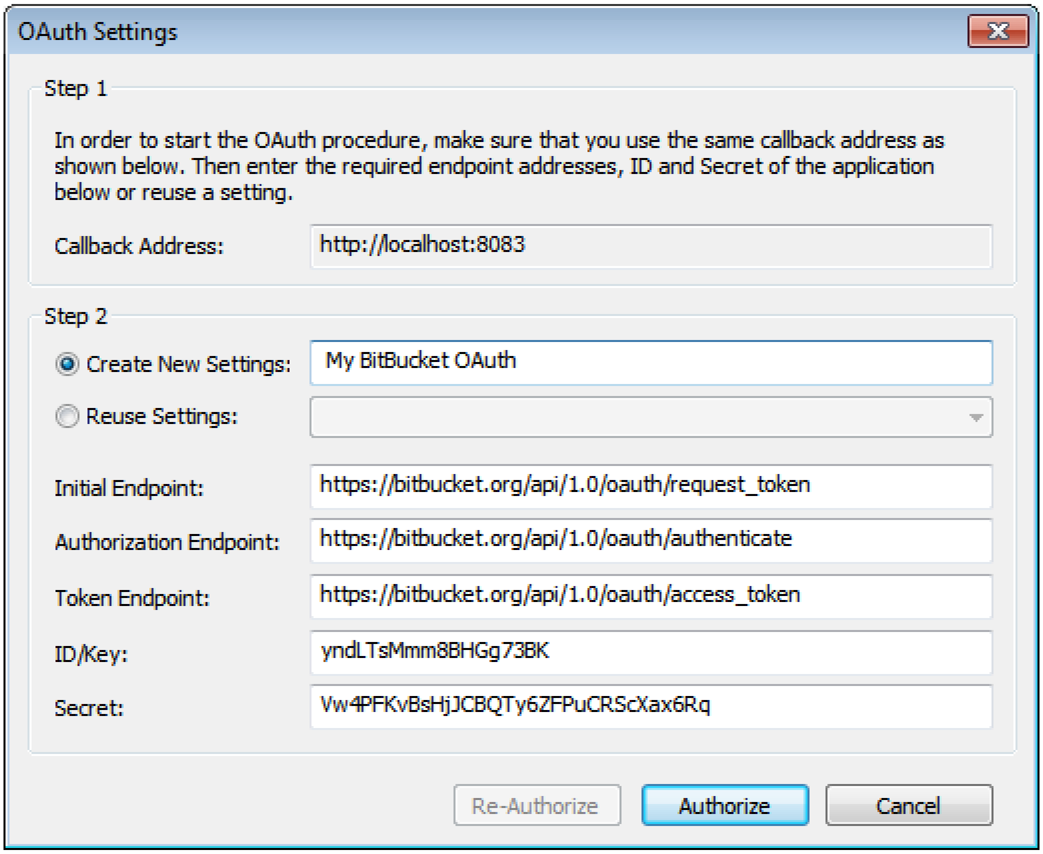
*OAuth authentication*
OAuth essentially authenticates MobileTogether Designer and authorizes access to the resources of the web service identified by the URL. This assumes that the web service supports OAuth. As an example web service that supports OAUTH, see the BitBucket documentation about OAuth 1.0.

The OAuth system works broadly as follows:

1. At the web service, create an OAuth key (or ID) and secret. Together these are known as an OAuth consumer.
2. Make a note of the web service's OAuth endpoints. There are three endpoints for OAuth 1.0 (Initial Endpoint, Authorization Endpoint, and Token Endpoint), and two endpoints for OAuth 2.0 (Authorization Endpoint and Token Endpoint). The endpoints are usually constant for all consumers.
3. Set up the application to access the web service with these five (OAuth 1.0) or four (OAuth 2.0) pieces of authentication information.

After you have obtained your key and secret from the web service, and noted the endpoints you need, you are ready to set up MobileTogether Designer to access the web service. Do this as follows:

1. In the Authentication Settings dialog (*screenshot above*), click **Setup OAuth 1.0** or **Setup OAuth 2.0**. The OAuth Settings dialog (*screenshot below*) appears.



OAuth Settings

Step 1

In order to start the OAuth procedure, make sure that you use the same callback address as shown below. Then enter the required endpoint addresses, ID and Secret of the application below or reuse a setting.

Callback Address: http://localhost:8083

Step 2

● Create New Settings: My BitBucket OAuth

○ Reuse Settings:

Initial Endpoint: https://bitbucket.org/api/1.0/oauth/request_token

Authorization Endpoint: https://bitbucket.org/api/1.0/oauth/authenticate

Token Endpoint: https://bitbucket.org/api/1.0/oauth/access_token

ID/Key: yndLTsMmm8BHGg73BK

Secret: Vw4PFKvBsHjJCBQTy6ZFPuCRScXax6Rq

2. Set *Callback Address* to **`http://localhost:8083`**. This address is fixed; it is the address of the machine on which you are working.
3. *Create New Settings:* Give your settings a name. This enables the settings to be reused (via the *Reuse Settings* combo box option) for other solutions that use the same resources.
4. Enter the endpoints declared by the web service. These are usually constant across the web service for all consumers of the service.
5. Enter your key (or ID) and secret.
6. Click **Authorize** to finish.

## Parameters

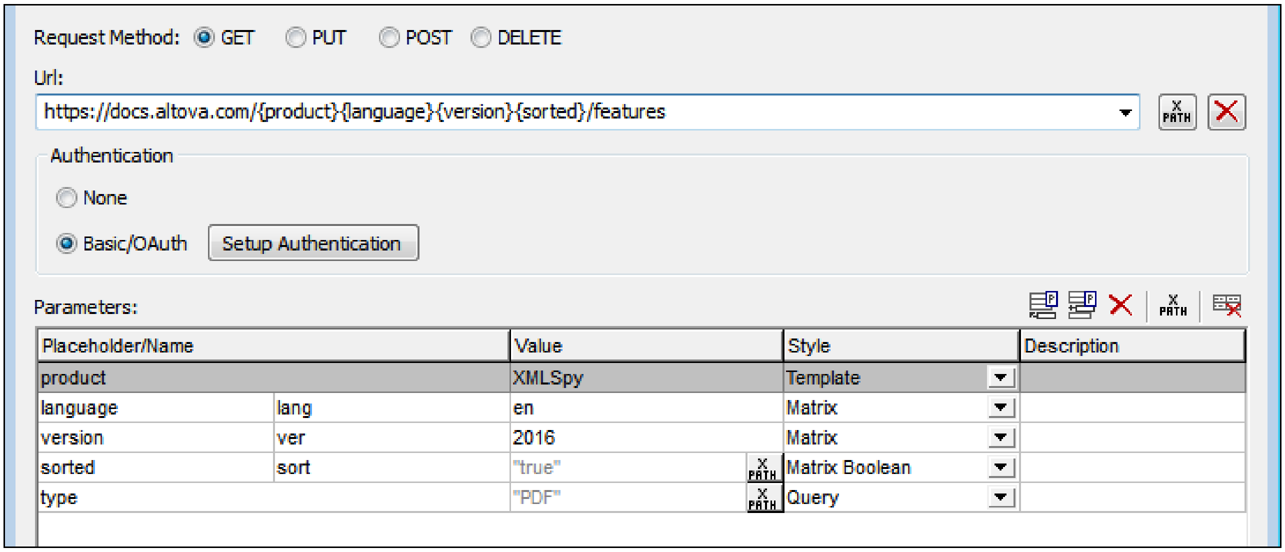
In the *Define Own Settings* template, parameters can be freely added, edited and deleted. In the *Use URL* and *Use WADL* templates, however, you can only edit the values of parameters; you cannot add or delete parameters, or edit their names.

You can add the following **types (or styles) of parameters** (*see the 'Styles' column of the screenshot below.*):

- *Template:* Template parameters use placeholders to substitute a value in a URL at runtime. For example, in the screenshot below, there is one template parameter with the placeholder `{product}`. This placeholder is used in the URL (*see screenshot below*). It has curly braces around it to indicate that it is a placeholder. When the URL is used at runtime, the value of the placeholder is substituted in the corresponding place in the URL. The relevant part of the URL would therefore resolve to: `https://docs.altova.com/XMLSpy.../features`.
- *Matrix:* In the case of matrix parameters, the placeholder in the URL is replaced by a `name=value` pair. In the screenshot example below, there are two matrix parameters, given in the URL by the

placeholders {language} and {version}. These placeholders in the URL would resolve to the parts highlighted in blue: `https://docs.altova.com/XMLSpy;lang=en;ver=2016.../features`. The semi-colon separator ; is prefxed to each parameter as part of the substitution.

- *Matrix Boolean:* If the value of a matrix boolean parameter is set to `true`, then the parameter's placeholder is replaced by the parameter's name. If the value is set to `false`, then the parameter's placeholder is replaced by the empty string (that is, by nothing). So in the example shown in the screenshot below, the matrix boolean placeholder would resolve to the highlighted part: `https://docs.altova.com/XMLSpy;lang=en;ver=2016;sort/features`. The semi-colon separator ; is prefxed to each parameter as part of the substitution.
- *Query:* Query parameters do not use placeholders. All the query parameters are gathered into a query string and this string is appended at runtime to the path part of the URL. For example, the URL shown in the screenshot below will resolve at runtime to this: `https://docs.altova.com/XMLSpy;lang=en;ver=2016;sort/features?type=PDF`. The question mark separator **?** is prefxed to the query string. Additional queries are prefixed by the ampersand separator **&**. So a query string with two queries would look like this: `?type=PDF&about=json`.



Request Method: ● GET ○ PUT ○ POST ○ DELETE

Url: https://docs.altova.com/{product}{language}{version}{sorted}/features

Authentication

○ None

● Basic/OAuth

Parameters:

| Placeholder/Name | | Value | Style | Description |
|---|---|---|---|---|
| product | | XMLSpy | Template | |
| language | lang | en | Matrix | |
| version | ver | 2016 | Matrix | |
| sorted | sort | "true" | Matrix Boolean | |
| type | | "PDF" | Query | |

### *Columns of the Parameters table*

The Parameters table has four columns. The use of the first three columns is explained in the description of the types of parameters given above. Notice that template, matrix, and matrix boolean parameters use placeholders. While template parameter placeholders are replaced by values, matrix and matrix boolean parameter placeholders are replaced by name-value pairs and names, respectively. Query parameters have no placeholders; their name-value pairs are appended to the path part of the URL. The *Description* column contains descriptions of the parameter for you, the MobileTogether Designer user.

### *Parameter table icons and table editing*

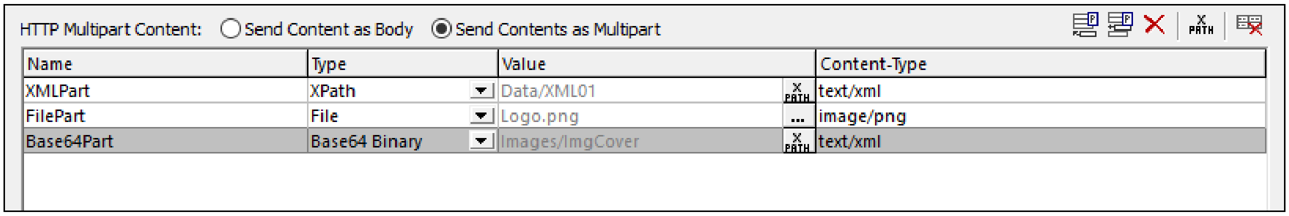
At the top right of the Parameters table, there are icons that enable you to manage entries in the table.

- *Appending and inserting parameters:* Use **Append** to add a new parameter as the last parameter in the table. Use **Insert** to insert a new parameter directly above the currently selected parameter. The order in which parameters are entered in the table is unimportant. It is the order of the placeholders in the URL that counts. All query parameters are gathered into a query string that is appended to the path part of the URL at runtime.
- *Deleting parameters:* Click **Delete** to delete the selected parameter.

- *XPath expressions for parameter values:* When a parameter is selected, click **XPath** to open the Edit XPath/XQuery Expression dialog and enter an expression that resolves to a string. This string is entered as the value of the parameter. In these cases, an **XPath** icon appears in the *Value* column of the parameter. Clicking this icon enables you to edit the XPath expression.
- *Resetting parameter values:* Click **Reset parameter** to delete the value of a parameter.
- *Editing parameter names and values:* Click in the respective field and edit.

## HTTP content

You can specify content to send with HTTP PUT and POST requests. You can send the content as a single item in the body of the request (*Send content as body*) or as multiple items in a multipart request (*Send contents as multipart*). Select the appropriate radio button from these two options.



HTTP Multipart Content: ○ Send Content as Body ● Send Contents as Multipart

| Name | Type | Value | Content-Type |
|---|---|---|---|
| XMLPart | XPath | Data/XML01 | text/xml |
| FilePart | File | Logo.png | image/png |
| Base64Part | Base64 Binary | Images/ImgCover | text/xml |

Append or insert a content entry using the icons at top right.

*Send content as body*

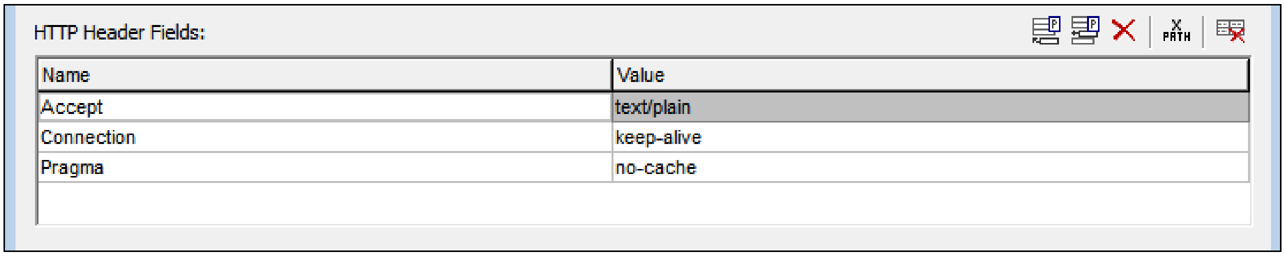
Content can be sent in the body (i) as a file, (ii) as content obtained by resolving an XPath expression, or (iii) as a Base64 string. For example, an image can be sent as a file or as Base64 content, and content can be entered directly or can be accessed (via an XPath expression) from an XML node.

*Send contents as multipart*

Enter the name for the content part being sent, the type of content access, the content itself, and the MIME type of the content. The content's value is the actual content being sent. In the screenshot above, the content is obtained, via XPath expressions, from page source nodes. Images are sent in Base64 format or as a file.

## HTTP header fields

HTTP header fields are colon-separated name-value pairs, for example: Accept:text/plain. Append or insert an entry for each header, and then enter the header name and value (*see screenshot below*).



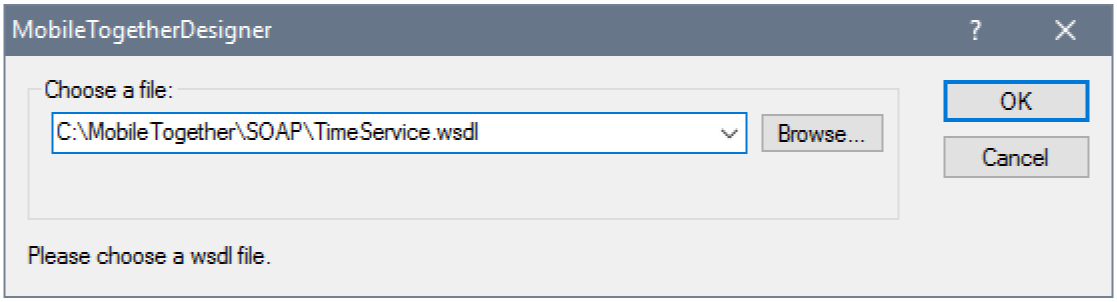
HTTP Header Fields:

| Name | Value |
|---|---|
| Accept | text/plain |
| Connection | keep-alive |
| Pragma | no-cache |

### 7.1.4.3 Via SOAP Requests

MobileTogether Designer enables you to make SOAP requests via WSDL. A WSDL file describes what operations are provided by a given web service. The SOAP protocol is then used to call one of these operations (over HTTP). The procedure in MobileTogether Designer for making the request is as follows:

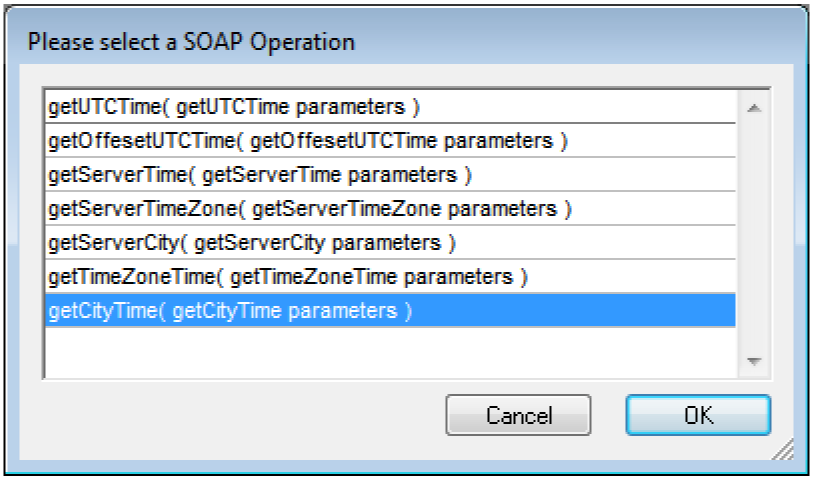
1. Add a page source[317] by clicking the **Add Page Source** icon in the Page Sources Pane. Select *New HTTP/FTP request with parameters*, and then select its *SOAP* radio button. Complete the page source settings in the next screen and click **Finish**.
2. A dialog appears that prompts you to browse for or enter a WSDL file (*screenshot below*). Select the WSDL file that defines the web service operation you want to request, and click **OK**.



Choose a file: C:\MobileTogether\SOAP\TimeService.wsdl

Please choose a wsdl file.

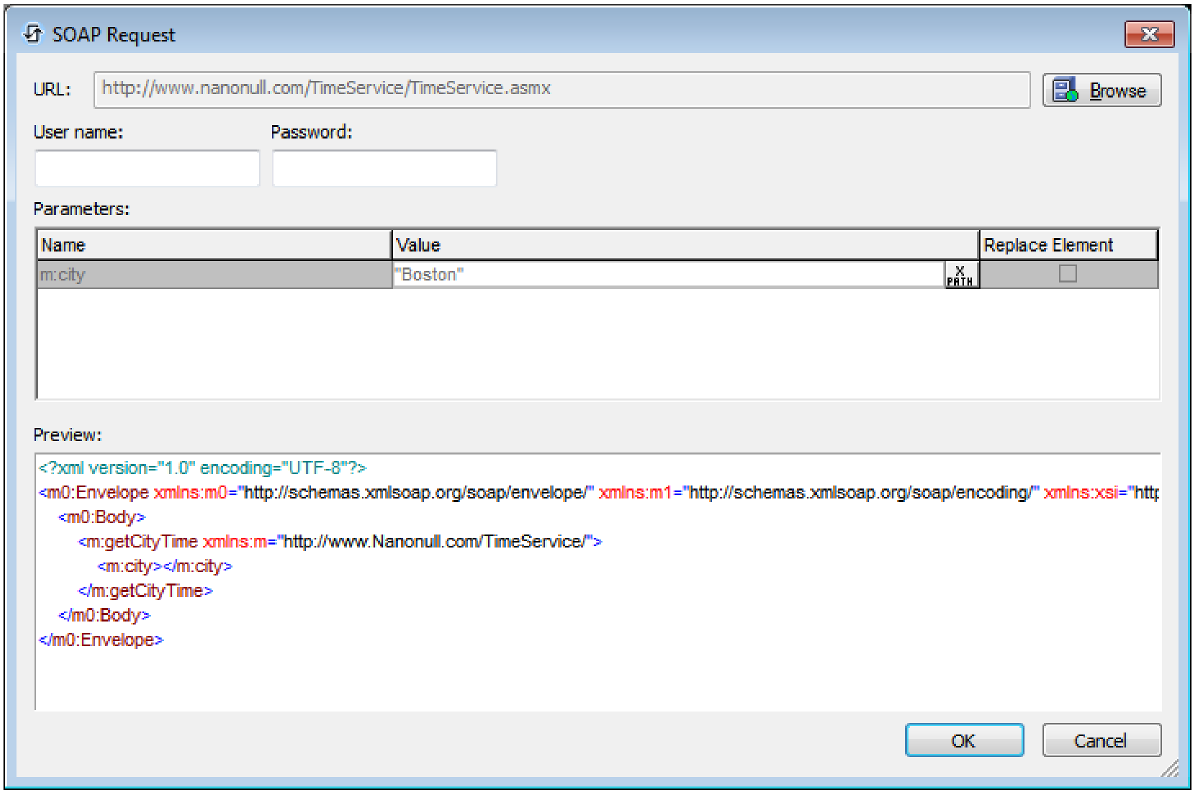
3. On clicking **OK**, the *Select a SOAP Operation (screenshot below)* is displayed. This dialog displays the web service operations described in the WSDL file. Select the operation you want to request, and click **OK**.



Please select a SOAP Operation

getUTCTime( getUTCTime parameters )
getOffesetUTCTime( getOffesetUTCTime parameters )
getServerTime( getServerTime parameters )
getServerTimeZone( getServerTimeZone parameters )
getServerCity( getServerCity parameters )
getTimeZoneTime( getTimeZoneTime parameters )
getCityTime( getCityTime parameters )

4. The SOAP Request dialog (*screenshot below*) appears. The URL in the *URL* field is the URL of the web service. The Preview pane shows the text of the SOAP request. If the request contains parameters, then these are listed in the Parameters pane, and you can enter an XPath expression that generates the value of the parameter. In the screenshot below, for example, the parameter `m:city` has been given a value that is generated by the XPath expression `"Boston"`. If you need to enter authentication information to access the web service, enter your user name and password in the respective fields. At the right of the URL field is a **Browse** button. Click it to choose another WSDL file and make another SOAP request.



URL: http://www.nanonull.com/TimeService/TimeService.asmx

User name:

Password:

Parameters:

| Name | Value | Replace Element |
|---|---|---|
| m:city | "Boston" | □ |

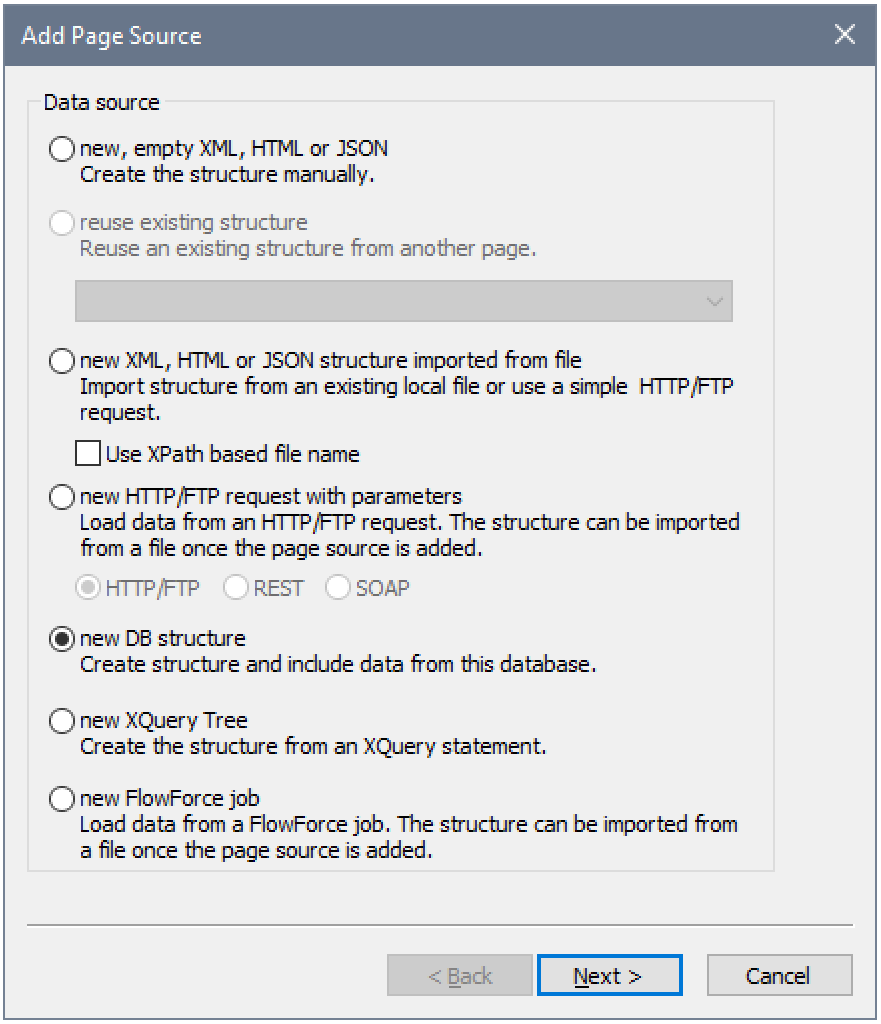
Preview:

```
<?xml version="1.0" encoding="UTF-8"?>
<m0:Envelope xmlns:m0="http://schemas.xmlsoap.org/soap/envelope/" xmlns:m1="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http
  <m0:Body>
    <m:getCityTime xmlns:m="http://www.Nanonull.com/TimeService/">
      <m:city></m:city>
    </m:getCityTime>
  </m0:Body>
</m0:Envelope>
```

5. Click **OK** when done. The SOAP request will be saved and will be sent at runtime.
6. Run a simulation to check the SOAP response.

## 7.1.5 DB Sources

This option enables you to create a page source structure from a database (DB) and to add data from that database. In the first screen of the Add Page Source dialog *(see screenshot below)*, select *New DB structure*.



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

○ new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

● HTTP/FTP ○ REST ○ SOAP

● new DB structure
Create structure and include data from this database.

○ new XQuery Tree
Create the structure from an XQuery statement.

○ new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

< Back    Next >    Cancel

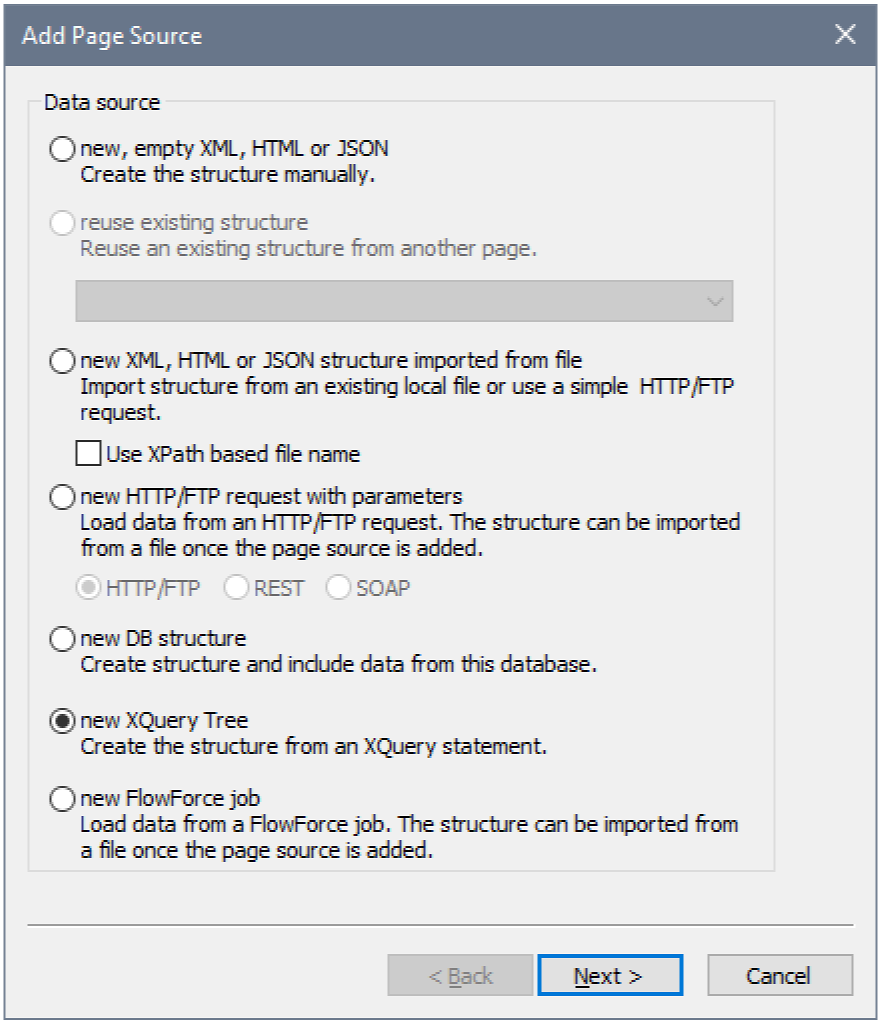
Click **Next** to go to the second screen of the dialog, where you can define the properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

On selecting this option and clicking **Finish**, the database (DB) Connection Wizard appears. After making the connection to the DB, you can select the table data to be imported for page source structure and data content. A root node (named **`$DB`** by default) is inserted, with the database name next to the name of the root node. You can change the name of the root node if you like by double-clicking in it to edit. The root node also has the DB structure below it. See the Databases[952] section for more information. The Database-And-Charts[159] tutorial provides an example of how to use DBs.

**Note:** If SQL statements are stored in a page source, they might trigger firewall rules while the design is executed on a client device. To prevent this, it is recommended that you do one of the following: (i) set the page source property *Keep data on* to `Server only`; (ii) use SSL for client connections; (iii) assemble the SQL statement on the server when required..

### 7.1.6 XQuery Sources

To add an XQuery page source, do the following: In the first screen of the Add Page Source dialog *(see screenshot below)*, select *New XQuery Tree*.



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

○ new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

● HTTP/FTP ○ REST ○ SOAP

○ new DB structure
Create structure and include data from this database.

● new XQuery Tree
Create the structure from an XQuery statement.

○ new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

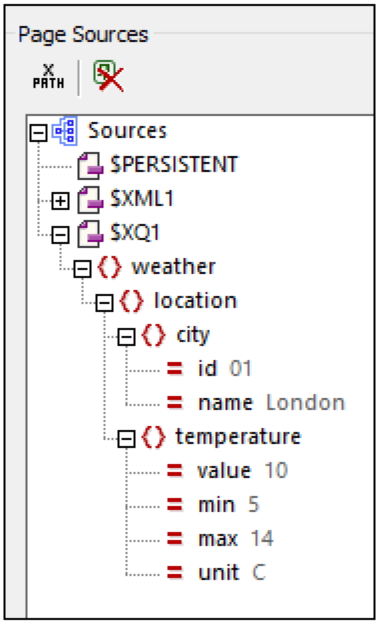
Click **Next** to go to the second screen of the dialog, where you can define the properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

On clicking **Finish**, the Edit XPath/XQuery Expression dialog[1244] will be opened. Enter an XQuery statement that generates the required data structure (and optionally data), and click **OK**. A page source with a root node named **$XQ** is created that has the structure specified in the XQuery statement. Right-click this root node, select the **Load Data** command and set the option to *On First Use* or *On Every Page*, as required.

For example, the following XQuery statement would generate the tree shown in the screenshot of the simulation further below.

```
element weather {
```

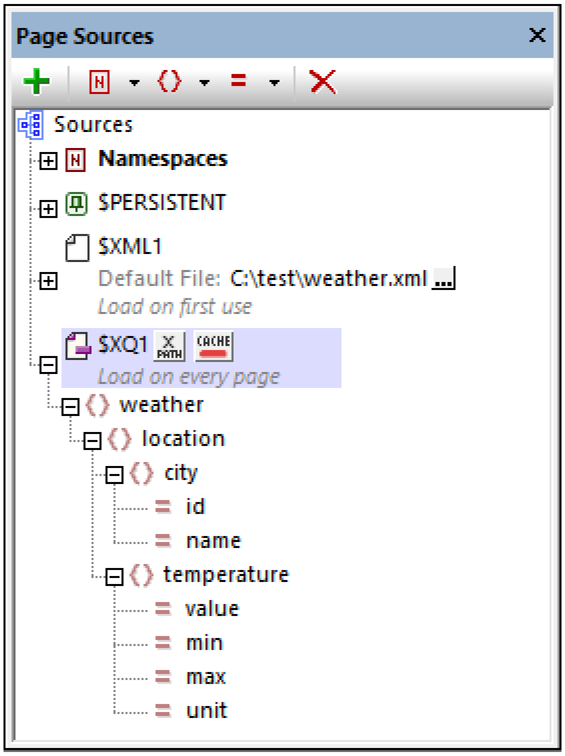
```
element location{
element city {attribute id{"01"}, attribute name{"London"}},
element temperature {attribute value{"10"}, attribute min{"5"}, attribute max{"14"},
attribute unit{"C"}}
}
}
```



Page Sources

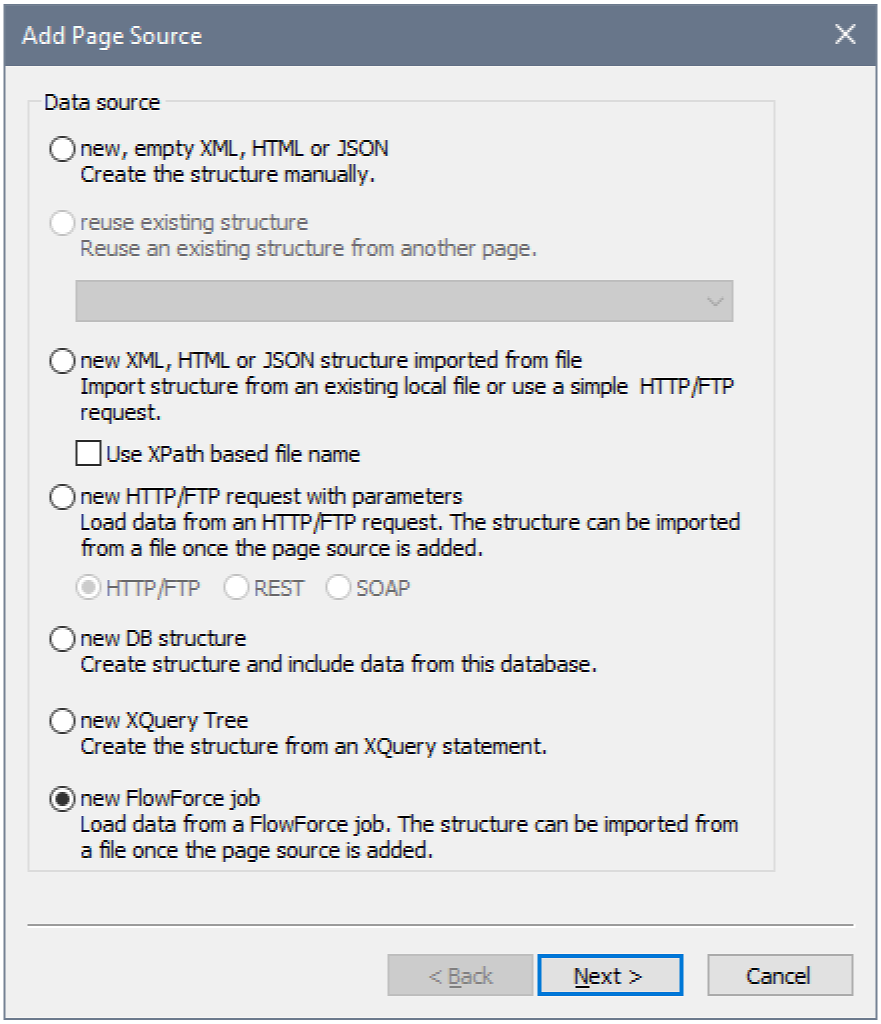
```
Sources
  $PERSISTENT
  $XML1
  $XQ1
    weather
      location
        city
          id 01
          name London
        temperature
          value 10
          min 5
          max 14
          unit C
```

If you want to use nodes from the **$XQ** tree in the design, you can locate them via XPath expressions (for example, like this: `$XQ1/weather/location/city/@name`). Alternatively, you can construct a temporary tree in the Page Sources pane 270 that matches the structure of the tree that will be created by the XQuery statement (*see screenshot below*); you can then drag nodes from the tree into the design. Note that the actual creation and loading of data into the tree will be according to the XQuery page source's **Load Data** option that you selected (*On First Use*, *On Every Page,* or *Not Automatically*).



## 7.1.7 FlowForce Jobs

You can add a page source that receives data from an Altova FlowForce job and takes its structure either from a file or a structure you create manually in the Page Sources Pane. To create such a page source, do the following: In the first screen of the Add Page Source dialog *(see screenshot below)*, select *New FlowForce job*.



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

○ new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

● HTTP/FTP ○ REST ○ SOAP

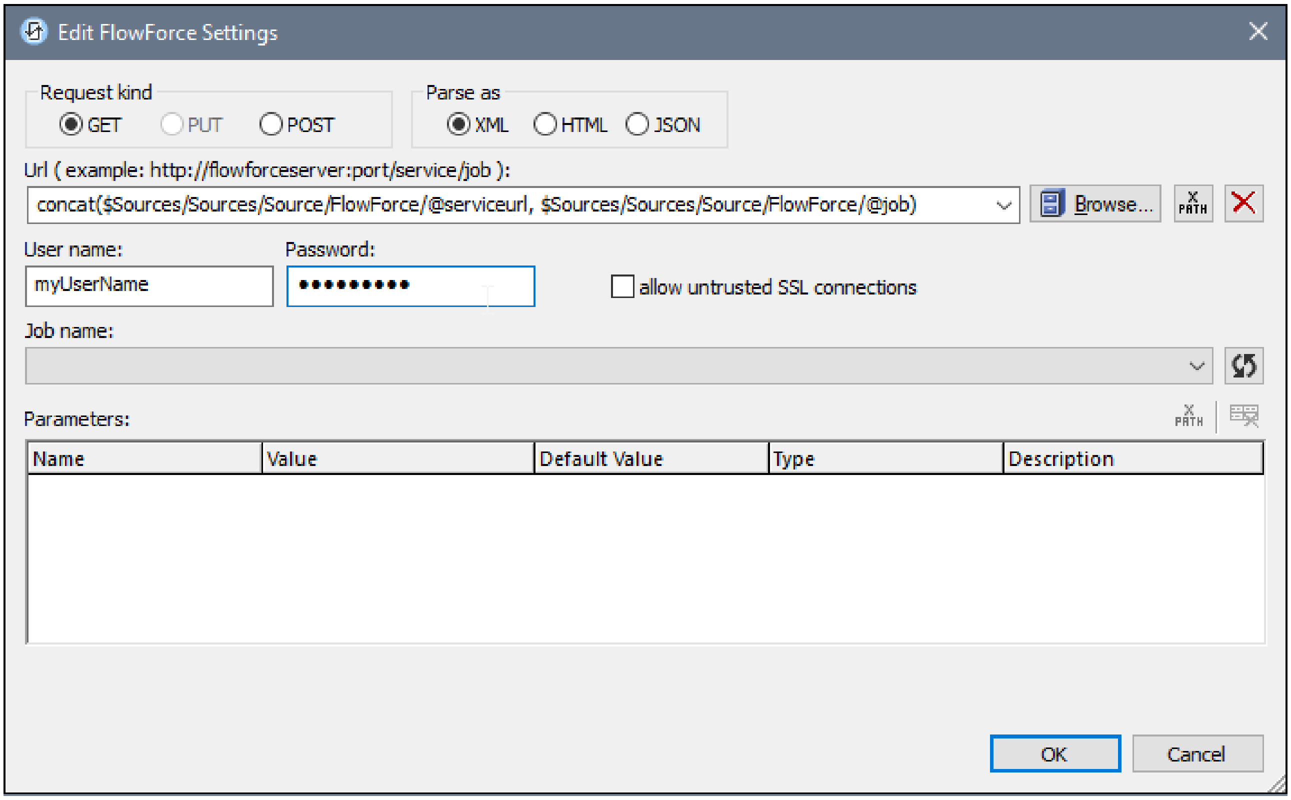
○ new DB structure
Create structure and include data from this database.

○ new XQuery Tree
Create the structure from an XQuery statement.

● new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

Click **Next** to go to the second screen of the dialog, where you can define the properties of the new page source. Here, you (i) specify the data type of the page source (XML, HTML, or JSON) and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

On clicking **Finish**, the Edit FlowForce Settings dialog *(screenshot below)* opens. Enter the FlowForce settings to connect to a FlowForce server and service. If you enter a URL and then click the **Refresh** button of the Job name entry, a connection to the FlowForce Server will be made and the available jobs will be displayed. When you select a job, its parameters will be displayed and you can enter the desired parameter values. If, however, you wish to compose the FowForce Server URL using an XPath expression, you must specify the job name in the URL and, in the *Parameters* pane, manually enter the parameters and their values.



Edit FlowForce Settings

Request kind: ● GET ○ PUT ○ POST

Parse as: ● XML ○ HTML ○ JSON

Url ( example: http://flowforceserver:port/service/job ):

concat($Sources/Sources/Source/FlowForce/@serviceurl, $Sources/Sources/Source/FlowForce/@job)

User name: myUserName

Password: ●●●●●●●●●

□ allow untrusted SSL connections

Job name:

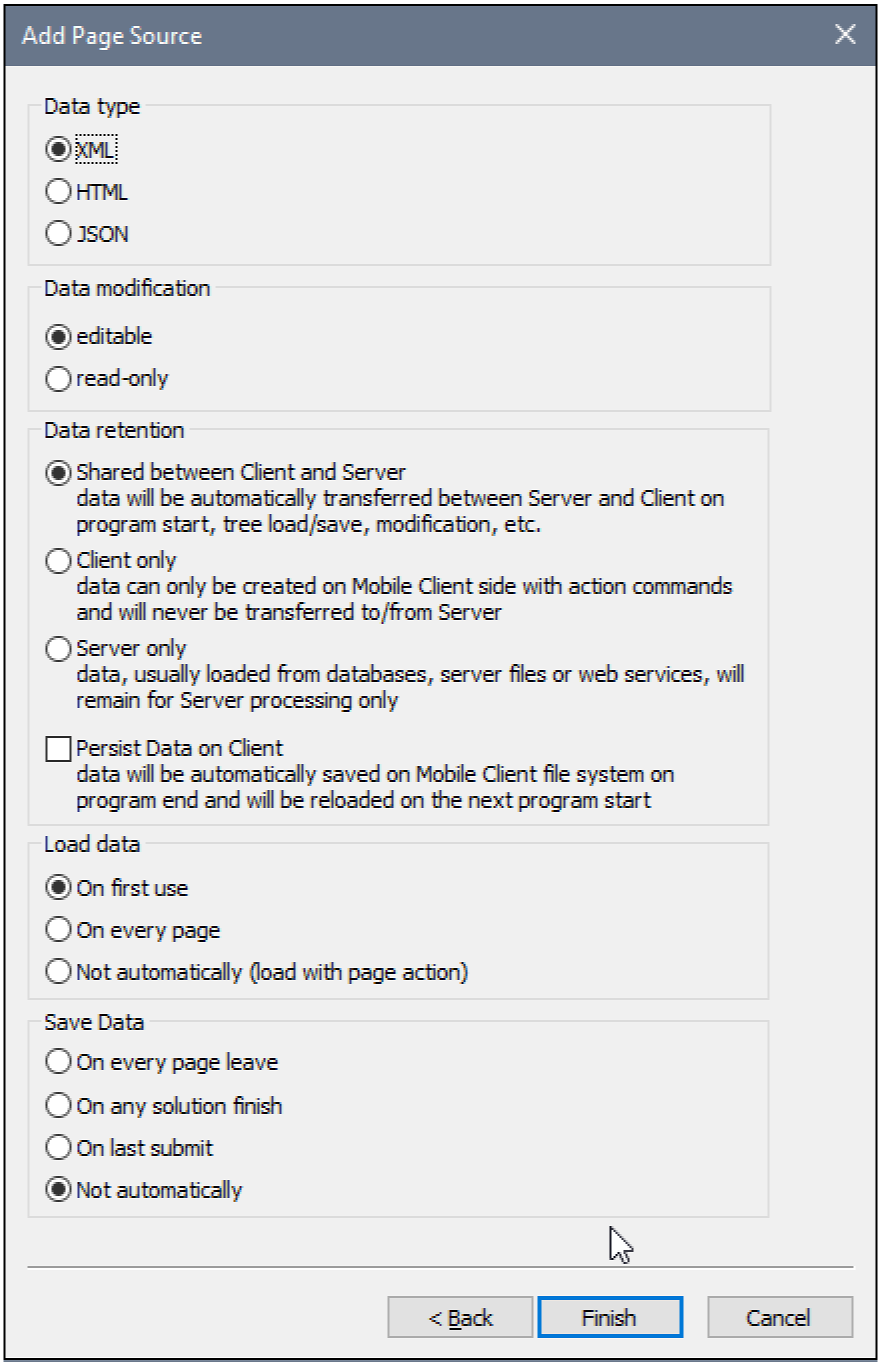
Parameters:

| Name | Value | Default Value | Type | Description |
|---|---|---|---|---|

On clicking **OK**, a new page source is created that contains data retrieved by running the FlowForce job. This data can be parsed as XML, HTML, or JSON, depending on the selection you made when defining the properties of the page source in the second screen of the Add Page Source dialog. The structure of the page source can be modified subsequently. How to do this is described in the section, Tree Structure 352. For information about Altova's FlowForce application, see the FlowForce page on the Altova website and the FlowForce product documentation.

## 7.2 Page Source Properties

After you have selected the type of page source that you wish to add 317, and clicked **Next**, the Add Page Source dialog displays the usage options of the selected page source (*see screenshot below*). Select the options you want, and then click **Finish**.



Add Page Source

Data type
● XML
○ HTML
○ JSON

Data modification
● editable
○ read-only

Data retention
● Shared between Client and Server
data will be automatically transferred between Server and Client on program start, tree load/save, modification, etc.
○ Client only
data can only be created on Mobile Client side with action commands and will never be transferred to/from Server
○ Server only
data, usually loaded from databases, server files or web services, will remain for Server processing only

□ Persist Data on Client
data will be automatically saved on Mobile Client file system on program end and will be reloaded on the next program start

Load data
● On first use
○ On every page
○ Not automatically (load with page action)

Save Data
○ On every page leave
○ On any solution finish
○ On last submit
● Not automatically

Note that some options will not be available for some types of page source. All the available options are listed below.

*Data type*
The format of data in the page source being added. Select from *XML*, *HTML*, or *JSON*. After the page source is added, you can change the data type in the context menu of the page source node (context command **Data Type** 359). The root node of the page source in all three cases is **`$XML`**. However, if you select *JSON*, then the root element will be named `json`. Note, however, that page sources that are read as JSON will also be saved as JSON (not as XML); this applies to saving via the **Save Data** 359 project property, and the Save action 803.

*Data modification*
Select *Editable XML* or *Read-only XML*. If a page source is created as editable, then data in its tree nodes can be modified. Data in read-only page sources cannot be modified. Both types of page source can be used to display data. But if you want to enable end users to write to data nodes, create the page source as an editable XML.

*Data retention*
Specifies whether data is: (i) copied to the client from the server, (ii) kept on the client, or (iii) kept on the server. There are two issues to consider when making this choice: (i) the calculations that are possible on client and server respectively, and (ii) how the location of the data can speed up processing.

Some calculations are done client-side only (for example, the resolving of XPath expressions to send an SMS); some calculations are done server-side only (for example, the creation of charts; only the final chart image is transferred to the client); and some calculations can be done both client-side and server-side (for example, the updating of XML tree nodes). All calculations are first attempted on the client. If a calculation is not possible on the client, the calculation is passed to the server. So, in order to save processing time, it is best to keep data where it can be accessed faster. If all calculations can be carried out client-side, then it is advisable to keep data on the client. Otherwise, you should consider one of the other two options. Note that chart data must not be saved on the client only; doing so will result in an error.

The *Persist Data on Client* option loads a solution with the client data it had when the solution was last closed. The client data is said to "persist" between two solution runs.

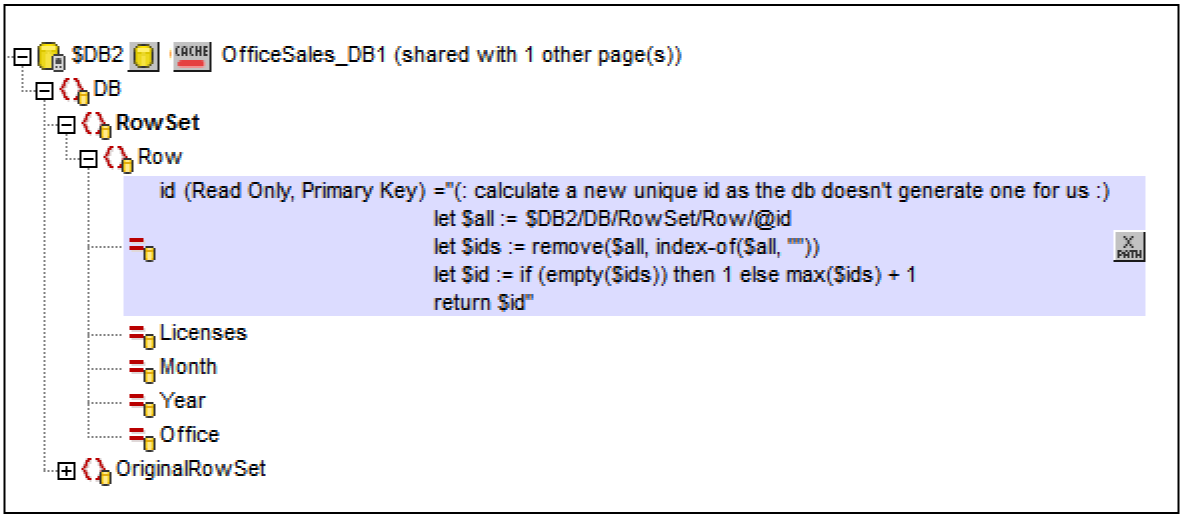
*Load data*
Selects whether data is loaded the first time the page is loaded, every time the page is loaded, or when specified by a page action. The selected option can be changed subsequently via the context menu of the source tree's root node.

*Save data*
This option is enabled if the page source is an external file (XML, HTML, or DB). It selects whether data is saved on (i) leaving the page, (ii) exiting the entire solution, (iii) when the user clicks the last **Submit** option, or (iv) when expressly defined as a page or control action. *See Page Source Actions* 799.

# 7.3 Page Source Trees

## Tree structure

A page source has an XML tree structure. The screenshot below shows the XML tree structure of a page source that is a DB table.



```
id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                               let $all := $DB2/DB/RowSet/Row/@id
                               let $ids := remove($all, index-of($all, ""))
                               let $id := if (empty($ids)) then 1 else max($ids) + 1
                               return $id"
```

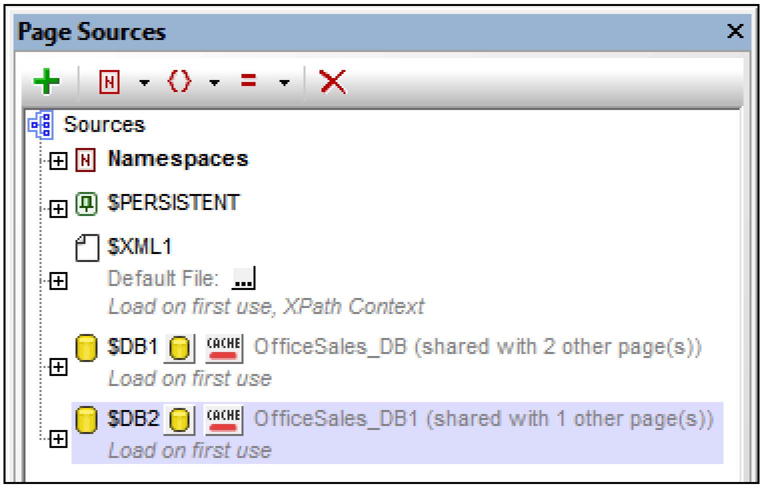
The structure of a page source is created in the following ways:

- Imported from an XML or HTML document, or an XQuery statement, when a page source is added (*see [Page Sources](317)*), or subsequently
- Created manually in the Page Sources Pane [270] by adding elements and attributes to a new empty XML tree (via the pane's toolbar icons [270])

## Accessing tree nodes

The nodes of every page source in the page can be accessed by XPath expressions anywhere in that page.

On each page, any node in any tree can be set as the **XPath context node for that page** (by right-clicking the node and selecting **Set as Page XPath Context**). All XPath expressions on the page will then be evaluated in the context of that node. The XPath context node of a page is indicated with the text *XPath Context*. In the screenshot below, **`$XML1`** is the XPath context node of the page. All XPath expressions on this page will be evaluated relative to **`$XML1`**.


Page Sources
Sources
Namespaces
$PERSISTENT
$XML1
Default File:
Load on first use, XPath Context
$DB1
OfficeSales_DB (shared with 2 other page(s))
Load on first use
$DB2
OfficeSales_DB1 (shared with 1 other page(s))
Load on first use

Whether a page context node is set or not, any node can be addressed by starting the locator expression with the root node of the specific tree. For example, in the XQuery statement that is highlighted in the first screenshot on this page, the second line has a `let` expression that locates the `@id` node with a locator expression that starts with the root node `$DB2`.

A **source node (or page source link)** is a tree node that is associated with a control.

- A source node is associated with control by dragging the source node from its tree onto the control.
- Once a source tree node has been defined as a page source link, it is displayed in a bold font in the page source tree.
- Typically, a page source link is used to display the source node's content in the control; for example, a Label control 554 would display the content of the associated page source link.
- In the case of charts and repeating tables, the control's source node serves as the context node (XPath origin) of all XPath expressions used to define properties of the control.

## Tree data

The data that is used in a MobileTogether solution is stored in the nodes of the project's page source trees. This data is obtained in one of the following ways:

- A file is specified as the default file of a page source. This file must have a structure that corresponds to the structure of the page source. Its data is then used as the data of the page source.
- A node can be given a fixed value (via the **Ensure exists (fixed value)** command in the node's context menu). This value overrides any value imported from a default file.
- A node is assigned an XPath expression (via the **Ensure exists (XPath value)** command in the node's context menu). The XPath expression generates the content of the node. This value overrides any value imported from a default file.
- A node is updated when an event is defined to trigger an *Update node* action, or when a node is the source node of a control that provides editing functionality (for example, the combo box and edit field controls).

## Page source links

A **source node (or page source link)** is a page source tree node that is associated with a control. In the cases of controls that provide editing functionality—such as the combo box and edit field controls—data edited by the end user is passed to the tree node. A source node is assigned to a control by dragging it from the Page Sources Pane[270] onto the control.

The page source link of a control is **displayed in bold** in the page source tree. When you hover over a page source link, a popup provides information about the associated control/s in the design. Conversely, controls that are associated with a page source link have an icon at the control's top left. Hovering over the icon displays information about the associated page source link.

## 7.3.1 Root Nodes

Each page source is conceptualized as a tree. The **root node** of each tree is identified by a **variable name** that is unique within each MobileTogether design (project). The terminology used to address and describe the basic parts of page source trees is given below. A listing of all the page source variables of a project, together with their uses in controls and actions, can be displayed in the Messages window when you click the command **Refactor | List Usages of All Page Source Variables**[1610].

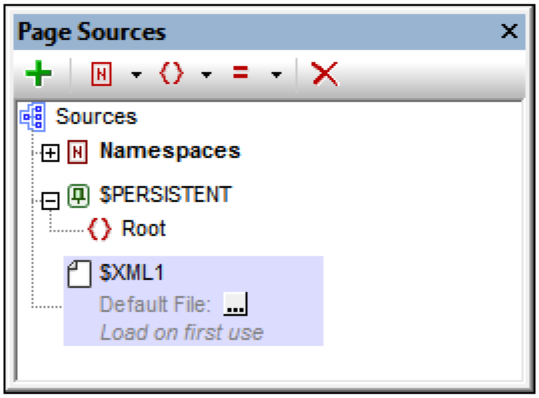
| | |
|---|---|
| **`/ = $RootNodeName`** | • The **root node** is the topmost node of a tree.<br>• A root node is represented by a variable of the form **`$RootNodeName`**, where `RootNodeName` is a name that identifies a particular tree's root node. An example is **`$XML`**. *See Names of Root Nodes[349] below.* |
| **`<RootElement>`**<br>**`<Element-1/>`**<br>**`...`**<br>**`<Element-N/>`**<br>**`</RootElement>`** | • The **root element** is the outermost element of an XML document, and contains all the other elements of the document. (In XML language, the root element is also sometimes called the **document element**.)<br>• The root element is considered to be a child of the root node. |

### Accessing tree nodes with XPath

The nodes of a tree can be accessed with XPath expressions. When using an XPath expression, be aware of what the context node is. One node across all the page sources of a page can be set as the XPath Context Node of the page (by right-clicking that node and selecting **Set as Page XPath Context**). This node will be the context node for all XPath expressions on the page. Exceptions are some controls, such as the chart[1149] and repeating table[1065] controls, which use their respective page source links as context nodes for XPath expressions that are used within the control. Whether a page XPath context node is set or not, a node in any tree is always accessible by starting the XPath expression with the tree's root node. For example: **`$XML1/root/element1`**.

### Names of root nodes

The names of root nodes are generated automatically when page sources are added[317]. The screenshot below, for example, shows two root nodes, named `$PERSISTENT` and `$XML1`. The name of a root node can be changed by double-clicking it and editing it.



The automatically generated name of the root node depends on the type of the page source: XML, HTML, HTTP-accessed, DB, XQuery, or FlowForce job

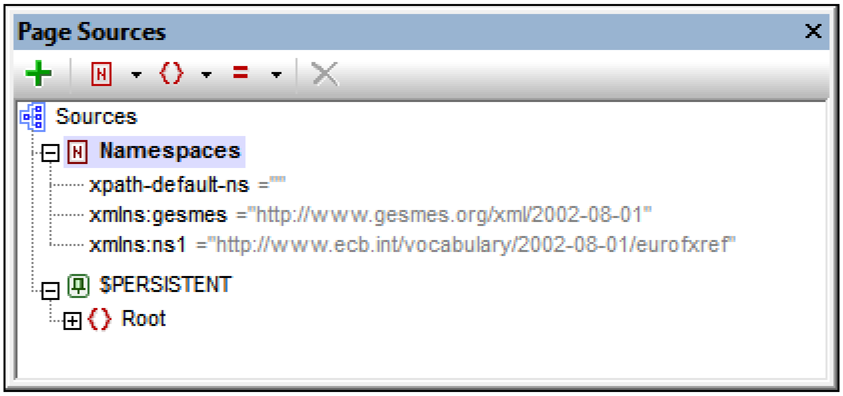
| | |
|---|---|
| **`$PERSISTENT`** | The `$PERSISTENT` tree is the tree saved on the client.<br>• This tree is created in every new design with an empty root element called `Root` (*see screenshot above*). A structure must be created for it, either manually 352, or by importing an XML structure 352 via its context menu's **Import** command.<br>• Data can be assigned to the nodes of the `$PERSISTENT` tree, either static data or dynamic data (using XPath expressions or by assigning a `$PERSISTENT` tree node to a control).<br>• `$PERSISTENT` tree nodes that are assigned (as page source links) to a control will be updated on the client. This means that when a solution is loaded in the client, nodes that are assigned to a control will take their data from the `$PERSISTENT` tree—and not from other page sources. |
| **`$XML`** | XML documents, created manually or imported. Default data file 354 is optional. |
| **`$HTML`** | HTML documents, created manually or imported. Default data file 354 is optional. |
| **`$HTTP`** | Documents accessed via HTTP/FTP, REST, or SOAP 325. The requested resource provides the data 354. |
| **`$DB`** | The selected database table provides both structure and data. |
| **`$XQ`** | XQuery documents. |
| **`$FLOWFORCE`** | FlowForce jobs. |
| **`$MT_CONTACTS`** | Created when a Read Contacts 692 action is added. Filled with data from the client's address book. |

| | |
|---|---|
| **$MT_CALENDAR** | Created when an Access Calendar 692 action is added. Filled with data from the client's calendars. |
| **$MT_DATALOGICSCANNER** | Created when a Datalogic Connect 778 action is added. The page source tree receives data when a barcode is scanned by a Datalogic scanner. *See Barcode Scanners 1145.* |
| **$MT_DBSTRUCTURE** | Created when a DB Read Structure 867 action is added. Has a standard structure. Nodes are filled at runtime with data from the DB that is read by the action. |
| **$MT_EMBEDDEDMESSAGE** | Created to the design when the **OnEmbeddedMessage** 397 event is activated, or when an Embedded Message Back 924 action is added |
| **$MT_FILEINFO** | Created when a Get File Info 849 or Read Folder 847 action is added. Holds information about the file/s and/or folder/s that are specified in these actions. |
| **$MT_GEOLOCATION** | Created when the Start/Stop Geo Tracking 735 action or the Read Geo Data 736 action is added to design. |
| **$MT_IN_APP_PURCHASE** | Created when the first In-App Purchase action 931 is added to the design. It stores app store data about products and purchases. See In-App-Purchase Page Source 1507 for details. |
| **$MT_MQTT** | Created when a Subscribe to MQTT Topic 764 action is added to the design. The page source will store the data of MQTT messages received. See Actions on Message Received 1138. |
| **$MT_NFC** | Created when NFC Start/Stop 746 action is added. Filled with data from last discovered NFC tag. |
| **$MT_PUSHNOTIFICATION** | Created at design time when an action is added to the **OnPushNotificationReceived** event. Filled with data from the payload of a received push notification (PN). *See PNs: The Receiving Solution 1128.* |
| **$MT_SERVICE** | Created when a service design is created 1545. It holds run time data about service triggers. |
| **$MT_ZEBRAMOBILECOMPUTER** | Created when a Zebra Mobile Computer Connect 775 action is added. Receives data when a barcode is scanned by an Zebra mobile computer scanner. *See Barcode Scanners 1145.* |
| **$MT_ZEBRASCANNER** | Created when a Zebra Connect 772 action is added. The page source tree receives data when a barcode is scanned by a Zebra scanner. *See Barcode Scanners 1145.* |

**Note:** A root node can be renamed at any time in the design process by double-clicking its name in the Page Sources Pane and editing the name. All references to the old name in XPath expressions throughout the design will be changed to the new name.

## 7.3.2 Namespaces in the Project

Namespaces are important for correctly identifying nodes, and for correctly locating nodes with the use of XPath expressions. The *Namespaces* item in the Page Sources Pane 270 (*screenshot below*) contains all the namespaces that have been declared for the project, irrespective of what page is currently active in Page Design View 253.


Page Sources
Sources
Namespaces
xpath-default-ns =""
xmlns:gesmes ="http://www.gesmes.org/xml/2002-08-01"
xmlns:ns1 ="http://www.ecb.int/vocabulary/2002-08-01/eurofxref"
$PERSISTENT
Root

Namespaces can be declared in two ways:

- *Automatic declaration on data import:* When an external XML file is added as a page source, namespaces in the source are automatically imported into the design and declared for the scope of the entire project. They then appear under the *Namespaces* item in the Page Sources Pane 270 (*see screenshot above*). The namespace prefixes are automatically set to match the original prefixes if such matching creates no ambiguities in the design. Prefixes assigned in the namespace declaration are used in node names, and must be used in XPath expressions that are intended to locate these nodes in the page source.
- *User-defined:* You can also add namespaces by clicking the **Namespace** icon in the toolbar of the Page Sources Pane 270 (*screenshot above*). Being able to add your own namespaces to a project enables you to create nodes that belong to one or more user-declared namespaces. This is useful for disambiguating between nodes that have the same local name.

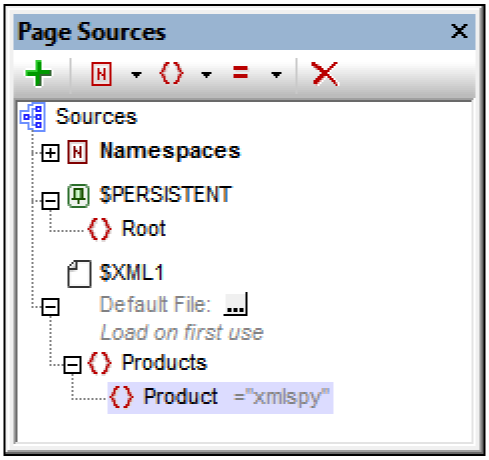
To delete a namespace, select it and click **Delete** in the pane's toolbar 270.

**Note:** A namespace prefix can be renamed at any time in the design process by double-clicking it in the Page Sources Pane and editing it. All references to the old prefix in XPath expressions throughout the design will be changed to the new prefix.

**Note:** The XPath default namespace (`xpath-default-ns=''`) is used for all XPath/XQuery functions, including extension functions and user-defined functions 1293.

## 7.3.3 Tree Structure

When a page source is added to a page 317, a root node 349 is created for that page source in the Page Sources Pane 270 (*screenshot below*). Depending upon the type of the page source 317 that is added, a tree structure might or might not be automatically created. For example, when a new XML tree is created with the structure being imported from an external XML file, a structure is created automatically at the time the page source is added. On the other hand, when a page source is added via an HTTP request[326], for example, no tree structure is created.



After a page source has been added and a root node has been created, a tree structure can be added in the following ways:

- By importing an XML structure from an external XML file
- By manually building up the tree using commands in the toolbar of the Page Sources Pane[270]

These methods are described below.

## Importing an XML structure

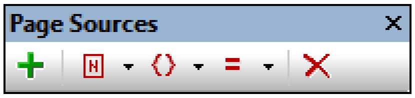
Right-click the root node in the Page Sources Pane[270], click **Import Structure from XML** in the root node's context menu, and browse for the XML file to use. The following outcomes are possible:

- If the root node has no descendant tree structure, the XML file's root element and its tree structure are imported. The XML file's root element is added as the root element of the page source.
- If the root node has a root element, then this root element and all its descendants are replaced by the root element of the XML file and its tree structure.

The imported tree structure can be modified manually, as described below, and data can be assigned to its nodes (described in the next section, Tree Data[354]).

## Manually creating a tree structure

Elements and attributes can be added relative to any node in a tree structure (including the root node[349]), and they can be deleted. Select a node in a page source, and click the appropriate toolbar command (*see toolbar screenshot below*). Temporary elements and attributes are intended to hold data used for calculations or data that for any other reason should not be saved to file. The data of temporary nodes is not saved.



| Icon | Command | Does this... |
|---|---|---|
| + | **Add Source** | Displays the Add Page Source dialog[317]. A root node[349] is created for the page source that is added. Only one child element can be added to a root node. |
| N ▾ | **Add Namespace** | Inserts or appends a namespace declaration under the *Namespace* entry. Edit the default prefix if you want, and enter a namespace. |
| {} ▾ | **Add Element** | Inserts, appends, or adds a child element relative to the selected node. |
| = ▾ | **Add Attribute** | Inserts, appends, or adds a child attribute relative to the selected node. |
| ✕ | **Delete** | Deletes the selected node. |

### Adding node content manually

You can manually add content to individual nodes via two commands in the context menu of the selected node:

- *Ensure Exists Before Page Load (Fixed Value):* A fixed string value is added as content of the node and displayed in the tree.
- *Ensure Exists Before Page Load (XPath Value):* An XPath expression provides the content of the node. The XPath expression and Edit XPath icon are displayed in the tree.

The node's content is generated before the page is loaded, and the page is passed with this node content to the client.

Note that content added manually in this way overrides content added via a default file or with the **Ensure Exists on Load**[371] commands.

## 7.3.4 Tree Data

### Editable and read-only data

Data in tree nodes can be editable or non-editable (read-only), depending on whether the tree is that of an editable page source or a read-only page source[345]. Whether a page source is editable or read-only is defined at the time the page source is added[317]. If you wish to change the editable/read-only definition, delete the page source and re-create it with the new definition.

Client actions can change the content of editable nodes. For example, if a combo box is associated with a node of an editable page source, the end user's combo box selection will be passed to the associated node and become its modified value. In the case of read-only page sources, the content of associated nodes is used for display purposes only. These associated nodes are known as **page source links**. A source node link is added to a control by dragging the tree node onto the control.

### Assigning data to page sources

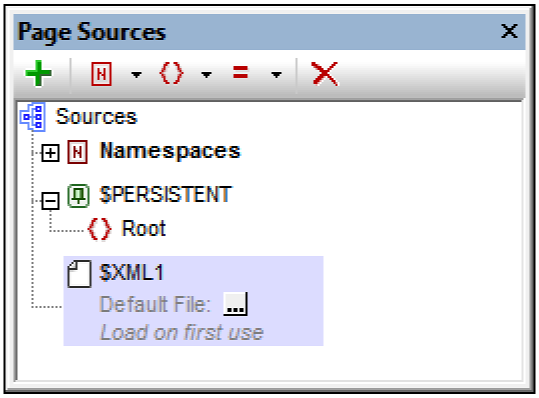
Data is assigned to nodes (in both editable and read-only data-source trees) in the following ways:

- *Assign a default file*[355]: The data in the default file is passed to the nodes of the tree and becomes the content of the nodes. The structure of the default file must be the same as that of the page source tree.

- *Add node content manually* 359: The context menu of every node has commands that allow the content of the node to be specified (the **Ensure exists** commands). If the node already has content assigned by another method (such as via a default file 355), the manually added node content 359 overrides the previously assigned content.

## Assigning a default file

An XML page source can have a default file assigned to it. The data in the default file will be passed to the page source as its data tree. To assign a default file do the following: Just below the root node name of the page source is an entry for the default file (*see screenshot below*).

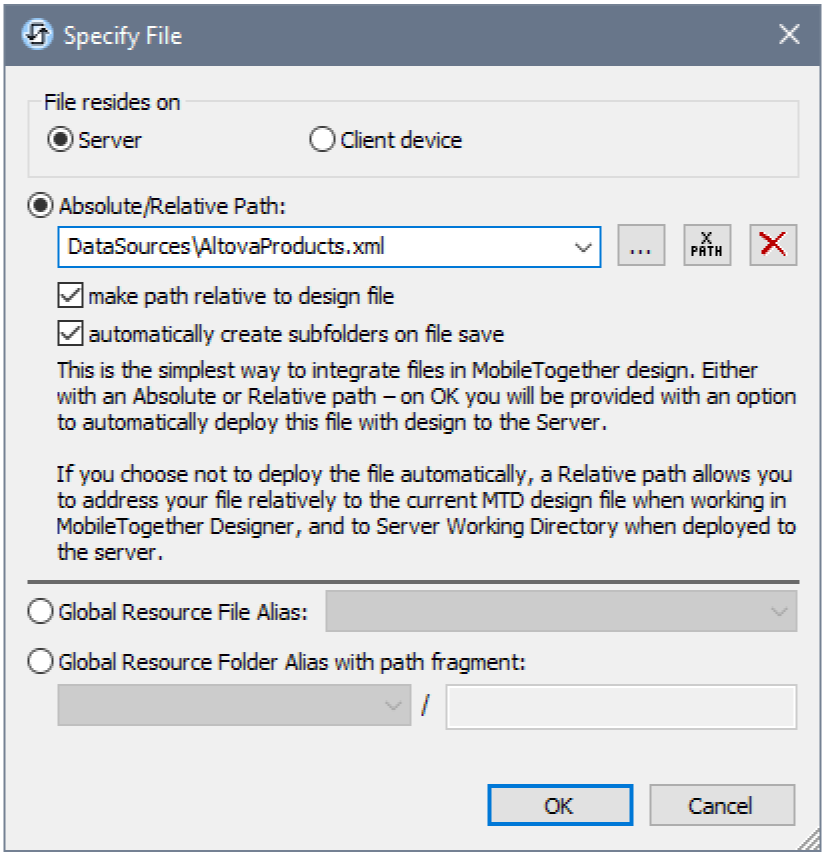


Click the **Additional Dialog** button to display the Specify File dialog (*screenshot below*), select the required file, and click **OK**. The assignment is made and the file path appears in the *Default File* entry. After a default file has been assigned, you can change the assignment by double-clicking the *Default File* entry and browsing for the new default file.

Data from the default file will be used as the data of the page source. However, in order for the data to be used, the default file must have the same structure as the page source. Note that, when a default file is assigned to a page source, its structure is not automatically imported. To import the structure of the XML file, use the context menu command **Import Structure from XML** 359. You can also manually create the structure of the page source 352 to match the structure of the default file.

*File is located on server*

If the default file is located on the server, select the *Server* radio button (*see screenshot below*). The dialog now enables you to browse for a file (*Absolute/Relative Path*) or to specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want.



File resides on

● Server ○ Client device

● Absolute/Relative Path: DataSources\AltovaProducts.xml

☑ make path relative to design file

☑ automatically create subfolders on file save

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:

○ Global Resource Folder Alias with path fragment: /

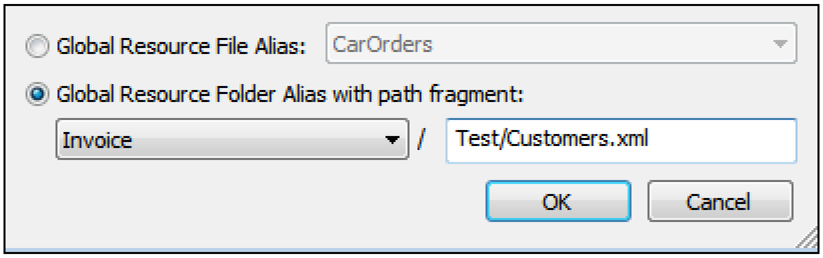
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). See the section Altova Global Resources[1334] for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).

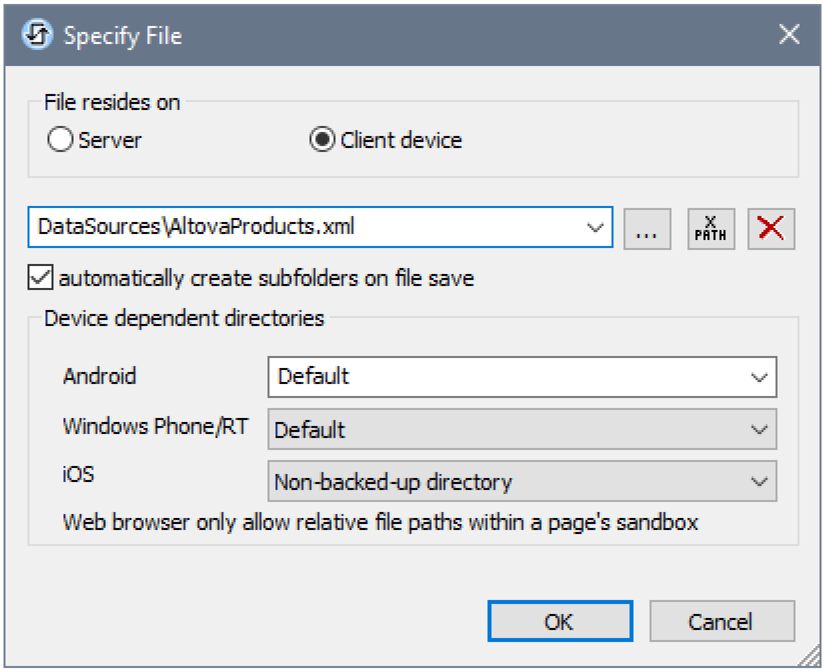


○ Global Resource File Alias: CarOrders
● Global Resource Folder Alias with path fragment:
Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources[1334] for details.

*File is located on client*

If the default file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



File resides on
○ Server ● Client device

DataSources\AltovaProducts.xml

☑ automatically create subfolders on file save

Device dependent directories

| | |
|---|---|
| Android | Default |
| Windows Phone/RT | Default |
| iOS | Non-backed-up directory |

Web browser only allow relative file paths within a page's sandbox

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To 686 (*Source File* and *Target File* options), Load/Save File 809, Load/Save Image 707, Load/Save Binary File 815, Load/Save Text File 821, Read Folder 847, and Get File Info 849.

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog 1663 (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).

- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

### Adding node content manually

You can manually add content to individual nodes via two commands in the context menu of the selected node:

- *Ensure Exists Before Page Load (Fixed Value):* A fixed string value is added as content of the node and displayed in the tree.
- *Ensure Exists Before Page Load (XPath Value):* An XPath expression provides the content of the node. The XPath expression and Edit XPath icon are displayed in the tree.

The node's content is generated before the page is loaded, and the page is passed with this node content to the client.

Note that content added manually in this way overrides content added via a default file or with the **Ensure Exists on Load** 371 commands.

## 7.3.5 Context Menus

Commands in the context menus of items in the Page Sources Pane 270 are described below. (The context menu of an item is accessed by right-clicking it.)

These commands are organized into two groups:

- Context menus of root nodes 359
- Context menus of any tree node 371.

### Context menus of root nodes

The commands listed below are available in the context menus of root nodes 349 (`$XML`, `$DB`, `$HTML`, etc). In addition to the commands that are common across all types of page sources (XML, DB, HTML, etc), some types of page sources have commands that are specific to its type (for example, commands for DB page sources). The specificity of such commands is noted where relevant.

▼ Insert, Append, Add Child

Enables the addition of elements and attributes relative to the selected node. **Insert** adds the node before the selected node. **Append** adds the node after the last node of that type. If you wish to add a node immediately after the selected node, go to the following node and use the **Insert** command.

▼ Delete, Delete from All Pages

Deletes the selected page source, respectively, from the current page and from all pages of the project.

▼ Keep data on

To reduce the amount of data transmitted over the mobile data network—which improves the performance of any mobile solution—MobileTogether lets you specify whether data from the selected page source is transmitted to client devices and/or kept on the server. For example, if a certain data set is only necessary to display a graph, then that data can be kept on the server. The image of the graph (say in PNG format) will be rendered by the server and transmitted to the client without the underlying data being transferred over the mobile network. For large data sets this produces a significant performance boost.

This command specifies whether data in the tree is stored on the server only, the client only, or shared by both. Note that if data is stored on the server, then it cannot be defined as persistent. See the **Persist data on client** command below.

▼ Read only data

A toggle command which specifies that data in the tree is read only. A read-only data tree is used to provide data for display and calculations. It cannot be used to hold data that needs to be edited.

▼ Persist data on client

A toggle command that makes a tree a persistent tree. Any number of trees can be defined as persistent. When a tree is defined as persistent, data in it is retained on the client device after the solution is exited. When the solution is opened the next time on that client, the persistent data is displayed. If a tree is defined as persistent, then it cannot be stored on the server. See the **Keep data on server** command above.

▼ Load data

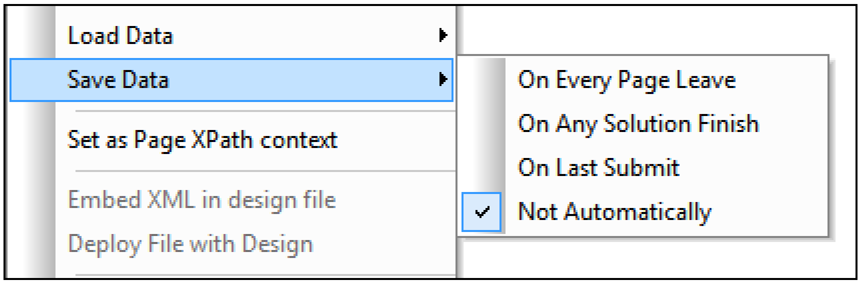
This command rolls out a sub-menu with the following mutually exclusive options (only one can be selected):

- *On First Use:* Loads the tree on entering a page where it is used. Once loaded, it will not automatically be reloaded anymore. If you share the same tree on multiple pages, then the first time one of such pages is opened (doesn't matter whether it is a Top page or Sub page), the tree will be loaded and will remain in memory.
- *On Every Page:* Reloads the tree every time you open a page containing the tree, whether the page is a Top page or Sub page. You must be careful with this option: It can slow the processing if loading takes significant time. But this will ensure that the data is retrieved afresh for every page.
- *Not Automatically:* The tree will not automatically be loaded for you. You must use the Reload (801), Load from File (809), or Load from HTTP (827) actions to load it. Alternatively, it can be created completely from scratch with the Append Node (875) and Insert Node (880) actions, without having to load data from any source. Note that you can use any of these five actions independently of the Load Data setting. That is, these actions can be used to reload your tree at specific moments even if **Load Data** is set to *On First Use* or *On Every Page*.

The default is *On First Use*.

▼ Save data

The **Save Data** command rolls out a sub-menu with the following mutually exclusive options (only one can be selected):



- *On Every Page Leave:* Data in the tree is saved every time a page containing that tree is exited.
- *On Any Solution Finish:* Data in the tree is saved when the solution is exited, no matter at what point or how the solution is exited.
- *On Last Submit:* Data in the tree is saved when the workflow progresses as designed, from first page to last page, and when the last **Submit** button is tapped. If this option is selected and the solution is exited before the last **Submit** button is tapped, then data in the tree will not be saved.
- *Not Automatically:* The tree will not automatically be saved. You must use the Save[801], Save to File[809], or Save to HTTP/FTP[827] actions to save data.

In the case of databases, there are options to save (i) modifications only; (ii) all rows; (iii) all rows if anything has been modified. If you wish to select an option that checks for modifications, then make sure that an **`OriginalRowSet`**[957] has been created for the table; otherwise an error will be reported on validation.

The default is *Not Automatically*.

▼ Reset data

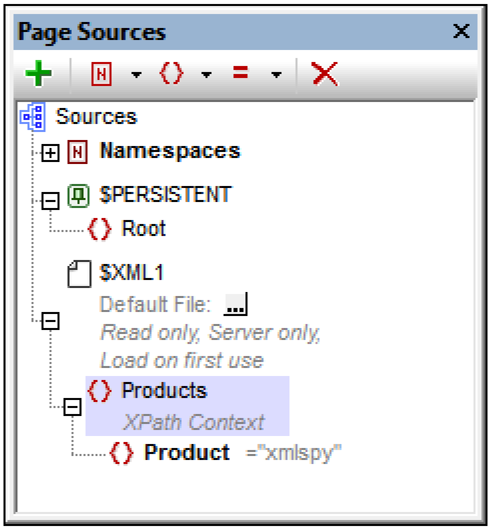
This command enables you to specify whether the data of the selected root node (page source) is automatically reset when a page is left. The following options are available:

- *On Every Page Leave:* The page source is reset on every page leave. Whenever a page is left, the page source will be reset—even if another page that is referencing the page source is still in use.
- *On Last Referencing Page Leave:* The page source is reset when the last page that references it is left. Essentially, a page source is reset only at the point when it is no longer in use by any page. Note that, if a sub page returns to a top page and both pages uses the same page source, then the page source will not be reset
- *Not Automatically:* The page source is not automatically reset when the page is left. If a data reset is required in a situation not involving a page-leave, then use the Reset[802] action.

The default is *Not Automatically*.

▼ Set as page XPath context

Sets the selected node as the XPath context node of the page. An annotation to the effect is displayed below the node (*see screenshot below*). The XPath context of the page is the context node for all XPath expressions on the page.

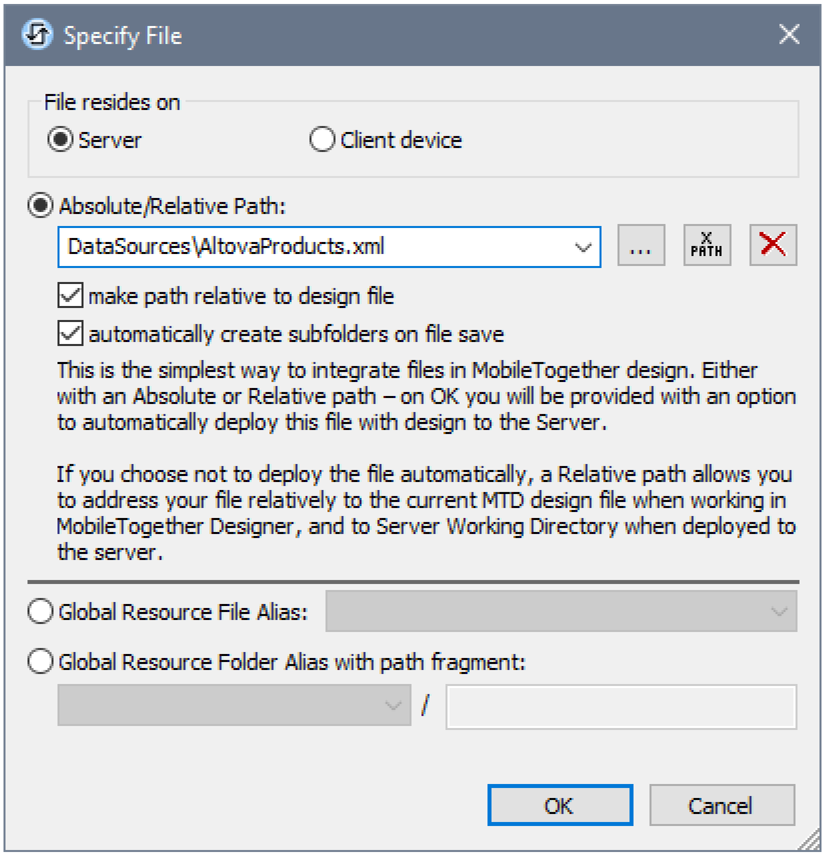


This command can be toggled on and off. So, you can enable a node as the XPath context node of the page, or you can toggle off a node's setting as the XPath context node of the page. If a node is set as the XPath context node when another node already has this setting, then the setting is toggled off for the previously assigned node and toggled on for the newly assigned node.

- Choose default file [root nodes with default files]

Displays the Specify File dialog (*screenshot below*), in which you specify the file to use as the default file. Data from the default file will be used as the data of the page source. However, in order for the data to be used, the default file must have the same structure as the page source. Note that, when a default file is assigned to a page source, its structure is not automatically imported. To import the structure of the XML file, use the context menu command **Import Structure from XML**; *see below*. You can also manually create the structure of the page source 352 to match the structure of the default file.

*File is located on server*
If the default file is located on the server, select the *Server* radio button (*see screenshot below*). The dialog now enables you to browse for a file (*Absolute/Relative Path*) or to specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want.



File resides on

● Server ○ Client device

● Absolute/Relative Path: DataSources\AltovaProducts.xml

☑ make path relative to design file

☑ automatically create subfolders on file save

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:

○ Global Resource Folder Alias with path fragment: /

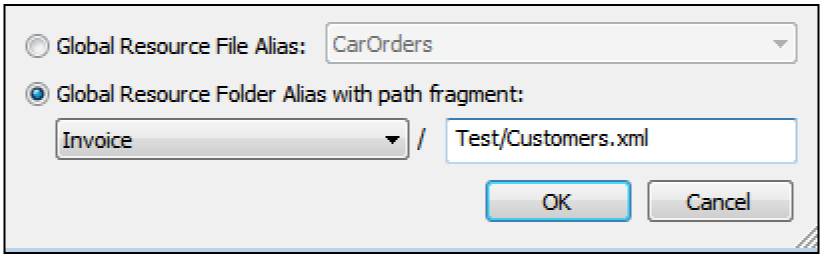
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). See the section Altova Global Resources 1334 for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

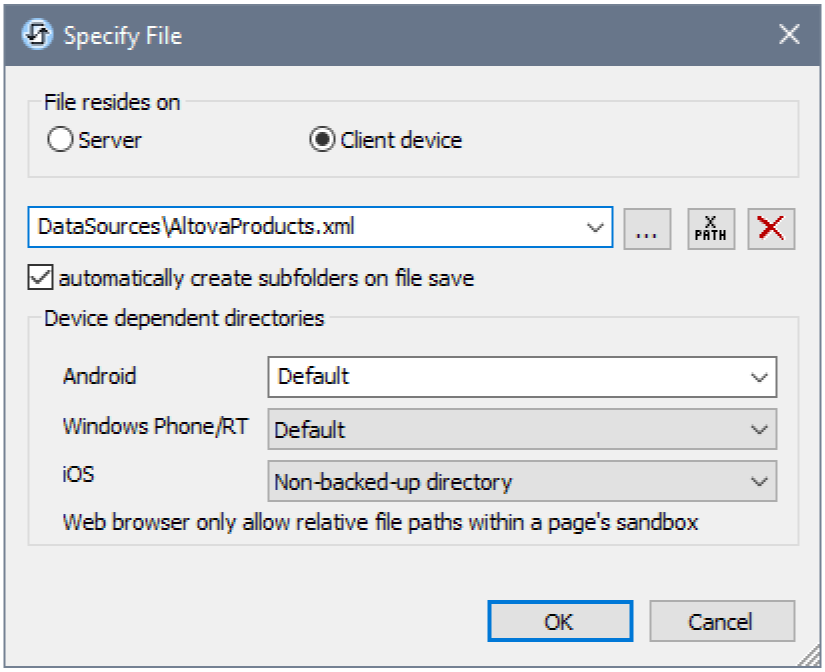
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 1334 for details.

*File is located on client*

If the default file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



Specify File

File resides on

○ Server ● Client device

DataSources\AltovaProducts.xml

☑ automatically create subfolders on file save

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To 686 (*Source File* and *Target File* options), Load/Save File 809, Load/Save Image 707, Load/Save Binary File 815, Load/Save Text File 821, Read Folder 847, and Get File Info 849.

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.
- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located

relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog[1663] (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).
- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.
- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

▼ View Default File in XMLSpy

This command is enabled when the root node[349] of an XML page source is selected that has a default file assigned to it[355]. Selecting it opens the default XML file in Altova's XMLSpy application, which enables you to work directly on the XML file while using the powerful editing and processing features of XMLSpy.

▼ Embed XML in design file

This command is enabled when the root node[349] of an XML page source is selected that has a default file assigned to it[355]. Selecting it embeds the XML data source in the design (`.mtd`) file. After a data source is embedded, the property *Embedded* is added to the annotation of the root node. See the sections

Location of Project Files[289] and Embed XML in Design File[312] for more information about (i) the advantages and disadvantages of embedding, and (ii) alternatives to embedding.

▼ Deploy file with design

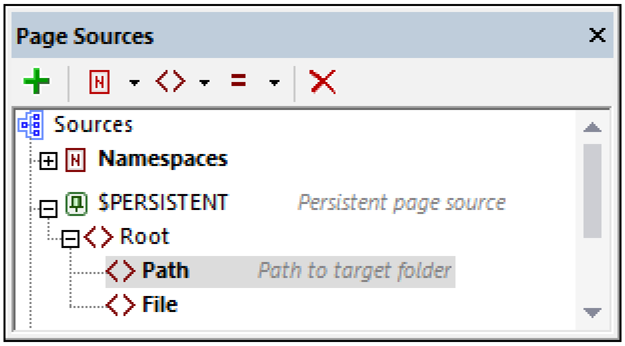
This toggle command is enabled when a page source is associated with a deployable file, typically a default file. The deployable file will already be listed in the Files Pane[259].

- Toggling the command on selects the check box of the file in the Files Pane[259], thus deploying it.
- Toggling the command off de-selects the file in the Files Pane[259]; the file will not be deployed.

Note that when a file is first added to the design, you are prompted about whether you wish to deploy the file or not.

▼ Comment

You can add comments to the root node of a page source tree and all tree nodes (*see screenshot below*). The comment is added to the right of the node. To edit a comment (or delete it), double-click the comment and edit it.



▼ List variable usage

The root node of every page source[349] is a variable, for example **$XML1** or **$DB1**. The **List Variable Usage** command lists, in the Messages Pane[278], all the usages of the selected root node variable. The items in the list are controls and actions in which the variable is used. (Variables are typically used in XPath expressions.) Clicking an item in the list highlights the control or opens the Actions dialog containing the variable usage.

▼ Data type

Select **XML**, **HTML**, or **JSON** from the submenu that rolls out. Your selection specifies what type of data source you plan to target, and enables MobileTogether Designer to correctly process the incoming or outgoing data. You can change this selection at any time. A change will cause the data source to be re-parsed for the new data type.

### Reload structure

Reloads the structure of the selected page source. The command is enabled only if the structure is based on an external resource, such as an XML file or DB. In the case of XML files, this means that if there is a default file[355], then the command is enabled.

If nodes have been manually added to the page source, then a dialog appears asking whether you want to remove or keep the added nodes. Click **Remove** to remove the selected nodes. You can deselect all nodes by clicking **Deselect User Added**, or you can deselect individual nodes. While the **Remove** button enables you to select which user-added nodes to remove and, conversely, which to keep, the **Keep All**. button keeps all user-added nodes, irrespective of whether they have been selected or deselected.

### Import structure from XML

Opens a Browse dialog in which you can select the XML or HTML file from which to import the XML structure of the page source tree. If the tree already contains a structure, you will be prompted about whether one or multiple nodes of the existing structure should be retained or not. If you choose to retain the existing structure and the new structure cannot be merged into the existing structure, then the new structure is imported as a sibling of the existing structure. This command is not available for tree structures that have a root element named `json`.

**Note:** When a structure is imported from an XML file, the file is set as the default file[355] and the file's data is also imported.

### Export structure to XML

Opens a Browse dialog in which you can select an XML file to which to export the XML structure of the page source tree. You can choose an existing XML file or create a new one. If you choose an existing file, the file's existing data will be overwritten by the exported structure. This command is not available for tree structures that have a root element named `json` and target a JSON data source[345].

### Import structure from JSON

Opens a Browse dialog in which you can select the JSON file from which to import the XML structure of the page source tree. If the tree already contains a structure, you will be prompted about whether one or multiple nodes of the existing structure should be retained or not. If you choose to retain the existing structure and the new structure cannot be merged into the existing structure, then the new structure is imported as a sibling of the existing structure. This command is available only for tree structures that have a root element named `json` and expect data from a JSON data source[345].

**Note:** When a structure is imported from a JSON file, the file is set as the default file[355] and the file's data is also imported.

### Export structure to JSON

Opens a Browse dialog in which you can select a JSON file to which to export the XML structure of the page source tree. You can choose an existing JSON file or create a new one. If you choose an existing file, the file's existing data will be overwritten by the exported structure. This command is available only for tree structures that have a root element named `json` and target a JSON data source[345].

▼ Choose DB data source [$DB only]

This command is enabled for database type (`$DB`) root nodes. It opens MobileTogether Designer's Database Connection Wizard, with which you can connect to a DB data source. After connecting to the DB, you can select the table to add as the page source. If the new table data cannot be merged into the existing structure, then the new table structure is created as a sibling of the existing structure.

If the DB is shared (as a page source) by other pages in the design, then you are prompted to choose from the following options:

- *Modify Shared Structure:* The modifications you are about to make to the page source structure on this page will be shared on the other pages where this DB structure is used.
- *Copy Structure:* The structure is copied to a new page source, and its root element is given a different name. The original page source is removed. The new data structure is not shared any more with any structure on other pages. You can now modify this page source while leaving page sources on other pages unchanged.
- *Cancel:* Cancels the modification process.

▼ Choose DB tables and views [$DB only]

This command is enabled for database type (`$DB`) root nodes. It opens MobileTogether Designer's Database Object Selector window, in which you can select the DB tables and views to import as a page source.

▼ Create OriginalRowSet [$DB only]

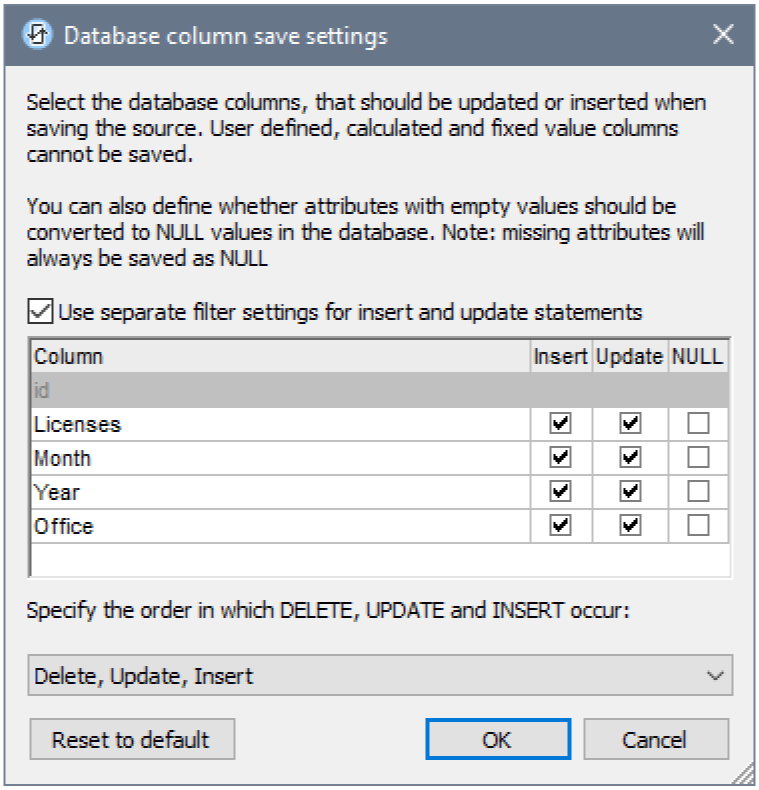
In order for data to be edited and saved, the tree of the page source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. The original data is saved in the `OriginalRowSet` element, while edited data is saved in the `RowSet` element. When the page source is saved, the difference between the two trees, `OriginalRowSet` and `RowSet`, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to `OriginalRowSet` so that `OriginalRowSet` contains the newly saved DB data, and the modification process can be repeated.

Note the following points:

- The `OriginalRowSet` element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command **Create OriginalRowSet**.
- The **Create OriginalRowSet**.command is enabled for database type (`$DB`) root nodes. It is a toggle command that creates/removes an `OriginalRowSet` data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the `OriginalRowSet` structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function **`mt-db-original-row`**[1262].

Filter columns [$DB only]

This command is enabled for database type (`$DB`) root nodes. It opens the Database Column Save Settings dialog, in which you can specify which columns should be saved to the DB data source.



Database column save settings

Select the database columns, that should be updated or inserted when saving the source. User defined, calculated and fixed value columns cannot be saved.

You can also define whether attributes with empty values should be converted to NULL values in the database. Note: missing attributes will always be saved as NULL

☑ Use separate filter settings for insert and update statements

| Column | Insert | Update | NULL |
|---|---|---|---|
| id | | | |
| Licenses | ☑ | ☑ | □ |
| Month | ☑ | ☑ | □ |
| Year | ☑ | ☑ | □ |
| Office | ☑ | ☑ | □ |

Specify the order in which DELETE, UPDATE and INSERT occur:

Delete, Update, Insert

The dialog displays the columns of the DB page source. You can specify which columns can be updated or can take inserted values. (Updates refer to modified data in existing row elements; inserted values refer to data in newly added row elements.) By default, the *Insert* and *Update* options of each column are selected together as a pair. If, however, you wish to specify different options for a column's *Insert* and *Update* options, check the *Use separate filter settings for Insert and Update statements* check box. Attributes with empty values can be converted to `NULL` values in the DB by checking that column's *NULL* check box. Note that missing attributes will always be saved as `NULL`.

Columns that cannot be updated (because they are user-defined, fixed-value, or calculated-value) will not have an *Insert*, *Update*, or *NULL* option check box. In the screenshot above, the `ID` column cannot be updated because it holds fixed values. Deselect the columns you do not want to update.

You can specify the order in which deletions, updates, and insertions occur by selecting the desired order in the combo box at the bottom of the dialog.

If you wish to reset the *Save settings* so that all columns are updated, click **Reset to default**.

Save Relation Settings

If the DB page source has related tables, then selecting this command opens a dialog in which the related tables are listed. Here you can select how data in each of these related tables should be handled when the DB page source is saved. The following options are available: (i) Only modifications in the related table will be saved; (ii) all rows of the modified table will be replaced; (iii) the related table will not be saved.

▼ HTTP/FTP Request Settings

This command is enabled for root nodes of the HTTP/FTP type 317 ( that is, `$HTTP` root nodes). Depending upon whether the current page source request is made with HTTP/FTP, REST, or SOAP, the appropriate settings dialog will be opened: Edit Web Access Settings 326, RESTful API Request Settings 328, Choose WSDL File 337.

▼ Cache settings

This command opens the Configure Caching settings dialog of the current tree. For a description of this dialog, see the section Data Sources (Data Sources) | Caches 376.

## Context menus of tree nodes

The commands listed below are available in the context menus of **tree nodes** (all elements and attributes except the root node). In addition to the commands that are common across all types of page sources (XML, DB, HTML, etc), some types of page sources have commands that are specific to its type (for example, commands for DB page sources). The specificity of such commands is noted where relevant.

▼ Insert, Append, Add Child

Enables the addition of elements and attributes relative to the selected node. **Insert** adds the node before the selected node. **Append** adds the node after the last node of that type. If you wish to add a node immediately after the selected node, go to the following node and use the **Insert** command.

▼ Ensure exists on load (fixed value)

This context menu item is available for tree nodes. A fixed value can be provided for the selected node when the page is loaded. Click the command and enter the value. This command is a toggle command. So, clicking it when a fixed value is already assigned will remove the value.

▼ Ensure exists on load (XPath value)

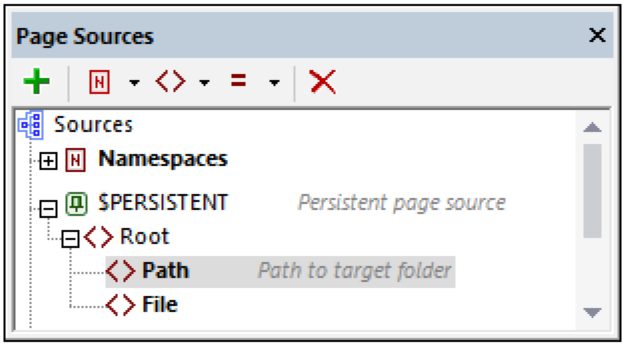
This context menu item is available for tree nodes. An XPath-generated value can be provided for the selected node when the page is loaded. On clicking the command, the Edit XPath/XQuery Expression Dialog 1244 is displayed. Enter the XPath expression to generate the value of the node. This command is a toggle command. So, clicking it when an XPath value is already assigned will remove the value.

▼ Is temporary

Sets the selected node as a temporary node. Data in temporary nodes is not saved when the tree is saved. Since temporary nodes are outside the framework of valid workflow data, they are intended for use in calculations and for storage of any data that is not wanted as part of the final data.

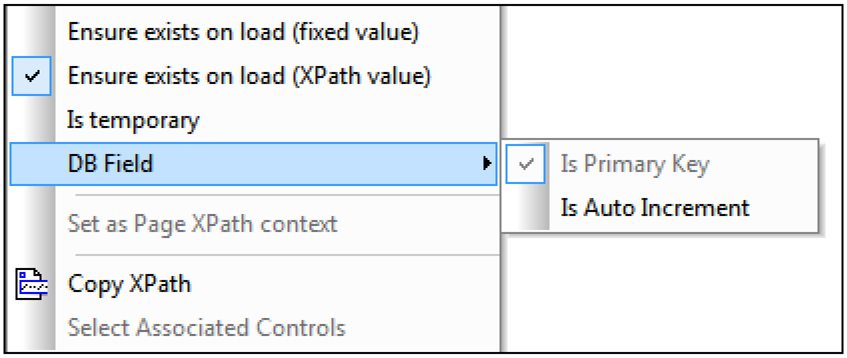
▼ Comment

You can add comments to the root node of a page source tree and all tree nodes (*see screenshot below*). The comment is added to the right of the node. To edit a comment (or delete it), double-click the comment and edit it.



▼ DB field

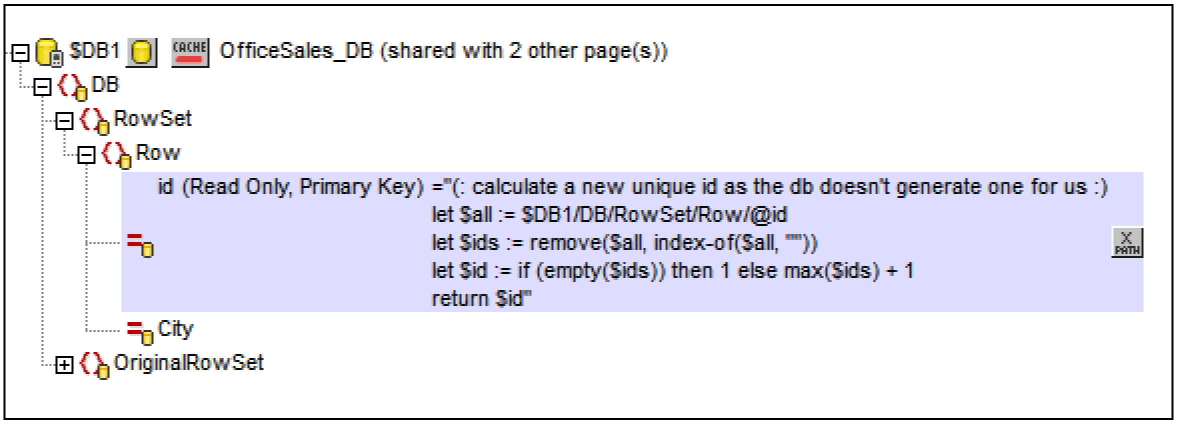
This context menu item is available for DB nodes, and has a sub-menu with two commands:


Ensure exists on load (fixed value)
Ensure exists on load (XPath value)
Is temporary
DB Field
Set as Page XPath context
Copy XPath
Select Associated Controls
Is Primary Key
Is Auto Increment

- *Is Primary Key:* Sets the selected node as the primary key column if a primary key has not already been auto-detected during retrieval from the DB.
- *Is Auto Increment:* Sets the selected node to auto increment. The node then becomes read-only.

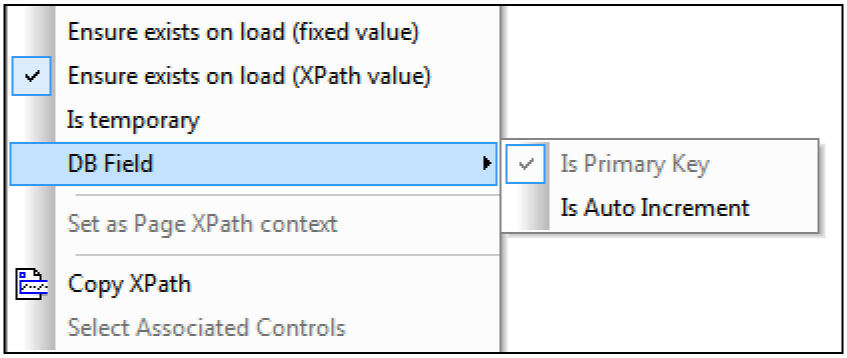
## Primary keys in MobileTogether Designer

Primary keys in DBs typically are auto-incrementing. When this is the case and a new row is added to a table, the primary key column of the added row is automatically incremented. In MobileTogether Designer, when a table is retrieved the primary key and auto-increment information is automatically retrieved and displayed in the Page Sources Pane (*see screenshot below*).



```
id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                              let $all := $DB1/DB/RowSet/Row/@id
                              let $ids := remove($all, index-of($all, ""))
                              let $id := if (empty($ids)) then 1 else max($ids) + 1
                              return $id"
```

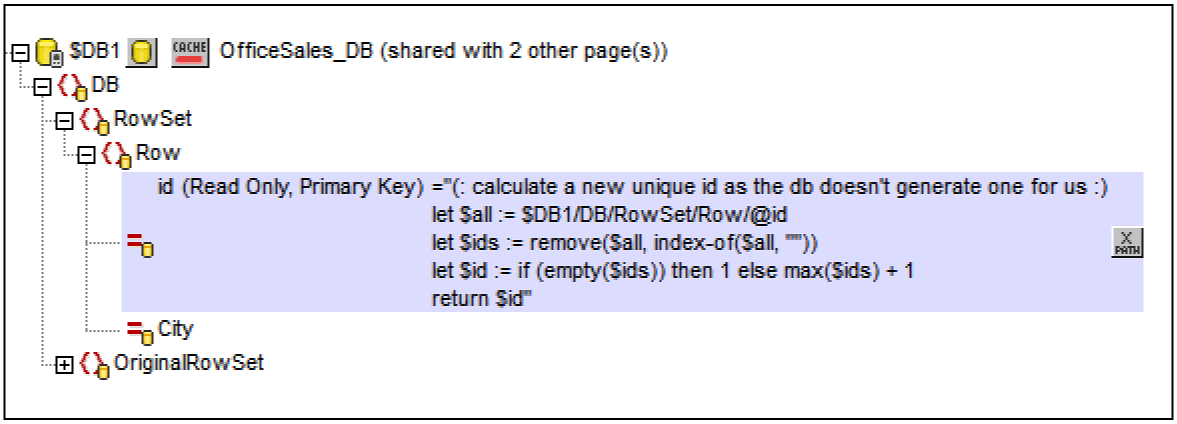
If auto-retrieval of this information was not successful, the context menu of tree nodes contains toggle commands that enable you to correctly annotate nodes (*see screenshot below*).



Ensure exists on load (fixed value)
✓ Ensure exists on load (XPath value)
Is temporary
DB Field
✓ Is Primary Key
Is Auto Increment
Set as Page XPath context
Copy XPath
Select Associated Controls

If the primary key column is not auto-incrementing, new primary key values for appended rows must be automatically generated using an XQuery expression. This is because primary key columns cannot be edited. The XQuery expression is inserted by using the primary-key node's context menu command, **Ensure Exists before Page Load (XPath Value)**. In the example below, a new value is generated for the primary key `@id` by using the following XQuery expression:

```
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```



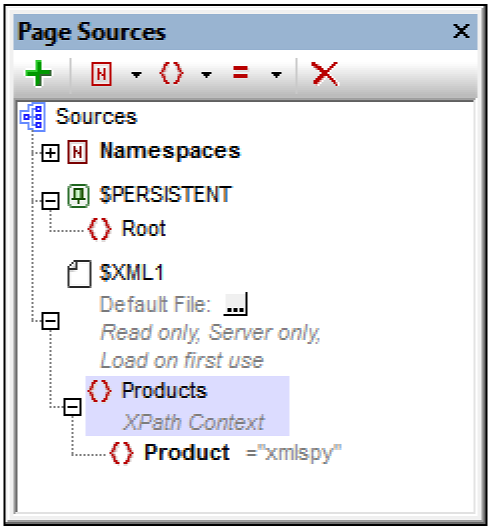
$DB1 OfficeSales_DB (shared with 2 other page(s))
DB
RowSet
Row

```
id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
    let $all := $DB1/DB/RowSet/Row/@id
    let $ids := remove($all, index-of($all, ""))
    let $id := if (empty($ids)) then 1 else max($ids) + 1
    return $id"
```

City
OriginalRowSet

## Set as page XPath context

Sets the selected node as the XPath context node of the page. An annotation to the effect is displayed below the node (*see screenshot below*). The XPath context of the page is the context node for all XPath expressions on the page.



Page Sources

Sources
Namespaces
$PERSISTENT
Root
$XML1
Default File:
*Read only, Server only,*
*Load on first use*
Products
*XPath Context*
**Product** ="xmlspy"

This command can be toggled on and off. So, you can enable a node as the XPath context node of the page, or you can toggle off a node's setting as the XPath context node of the page. If a node is set as the XPath context node when another node already has this setting, then the setting is toggled off for the previously assigned node and toggled on for the newly assigned node.

## Copy XPath

Copies the XPath locator expression of the node to the clipboard. The locator expression starts at the root node. For example: `$XML1Products/Product` is the locator expression of the `Product` node.

▼ Select associated controls

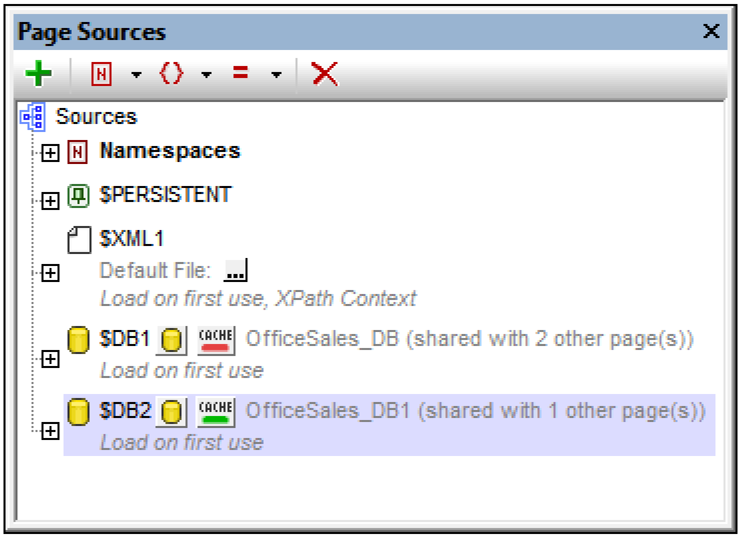
Selects the controls in the design diagram that are associated with the selected node. Such associations are typically page source links between the node and page controls.

# 7.4 Caches

If a page source gets its data from an XML file or DB, then that page source can be cached on the server. Such a page source has a Cache icon next to its name (*see screenshot below*). To create a cache for a page source, click the Cache icon, and, in the dialog that appears, configure the cache. A green Cache icon indicates that a cache has been created for the page source; a red icon indicates that no cache exists for the page source (*see screenshot below*). If a page source is **not** linked to an XML file or a DB, then it has no cache symbol (as for `$XML1` in the screenshot below).

There are two main reasons to create caches:

- If a page source generates reports slowly (for example, a large database)
- If a page source is not modified often. In both these cases, execution of a solution would be faster if data is taken from data caches on the server.


Page Sources
Sources
Namespaces
$PERSISTENT
$XML1
Default File:
Load on first use, XPath Context
$DB1
CACHE
OfficeSales_DB (shared with 2 other page(s))
Load on first use
$DB2
CACHE
OfficeSales_DB1 (shared with 1 other page(s))
Load on first use

Note the following points:

- If a page source is cached, then the cache will be used when the solution is run. This speeds up solution execution.
- A page source that is based on (i) a DB query that has **no** parameters, or (ii) an XML file will have one entry in its cache, which can be updated to contain the latest data in the external resource. Update times are specified in the cache configuration.
- A page source that is based on a DB query that has parameters can have multiple cache entries. Each cache entry corresponds to a different parameter combination. At configured update times, all defined cache entries will be updated.
- If a cache exists for a page source, then the cached data will be used when the solution is run.
- Caches can be used as soon as the solution has been deployed to the server.
- If you wish to not use cached data in a solution, you can deactivate the cache in the cache's configuration settings.
- You can also delete a cache in (i) the Cache Overview dialog of MobileTogether Designer, or the Cache Overview tab of MobileTogether Server.
- Once a cache has been defined in MobileTogether Designer, it can be used by the page sources of other designs, provided that the underlying data structure is compatible.

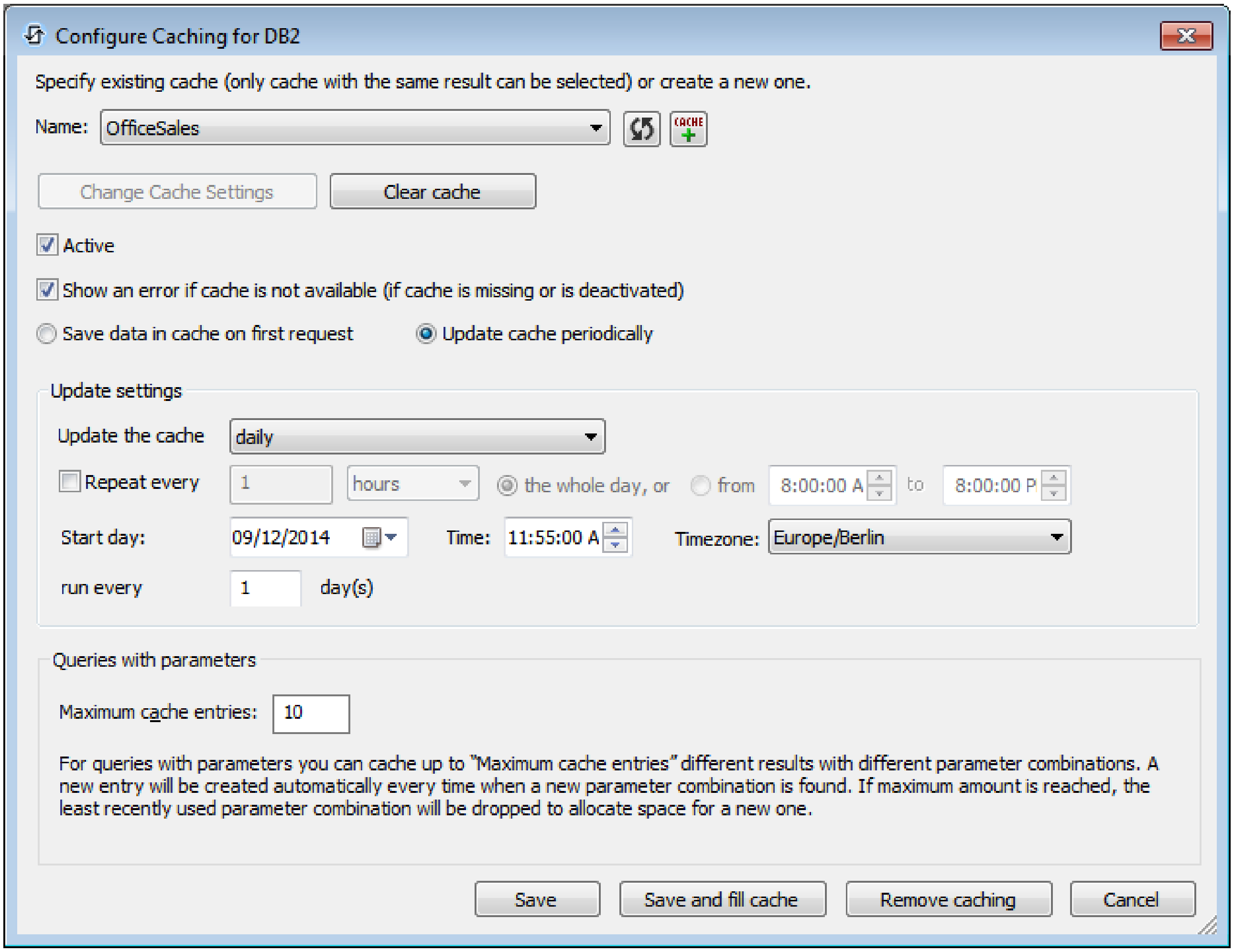
Working with caches involves two broad mechanisms, which are described in the sub-sections of this section

- Creating and configuring caches 377
- Managing and editing caches 378

## 7.4.1 Creating Caches

To create a cache for a page source, or to edit the settings of a cache that has already been created, click the **Cache** icon of the page source or select **Cache Settings** from the context menu of the root node of the page source. This displays the Configure Caching dialog (*screenshot below*). Set the properties of the cache as described below the screenshot, and click **Save** (to save the cache and its properties) or **Save and Fill Cache** (to save the cache as well as fill it with data from the data source).

All caches that have been saved to the server can be viewed in the *Cache* tab of the MobileTogether Server configuration page (web UI).



Configure Caching for DB2

Specify existing cache (only cache with the same result can be selected) or create a new one.

Name: OfficeSales

Change Cache Settings | Clear cache

☑ Active

☑ Show an error if cache is not available (if cache is missing or is deactivated)

○ Save data in cache on first request ● Update cache periodically

Update settings

Update the cache: daily

□ Repeat every 1 hours ● the whole day, or ○ from 8:00:00 A to 8:00:00 P

Start day: 09/12/2014 Time: 11:55:00 A Timezone: Europe/Berlin

run every 1 day(s)

Queries with parameters

Maximum cache entries: 10

For queries with parameters you can cache up to "Maximum cache entries" different results with different parameter combinations. A new entry will be created automatically every time when a new parameter combination is found. If maximum amount is reached, the least recently used parameter combination will be dropped to allocate space for a new one.

This dialog contains the following settings and controls:

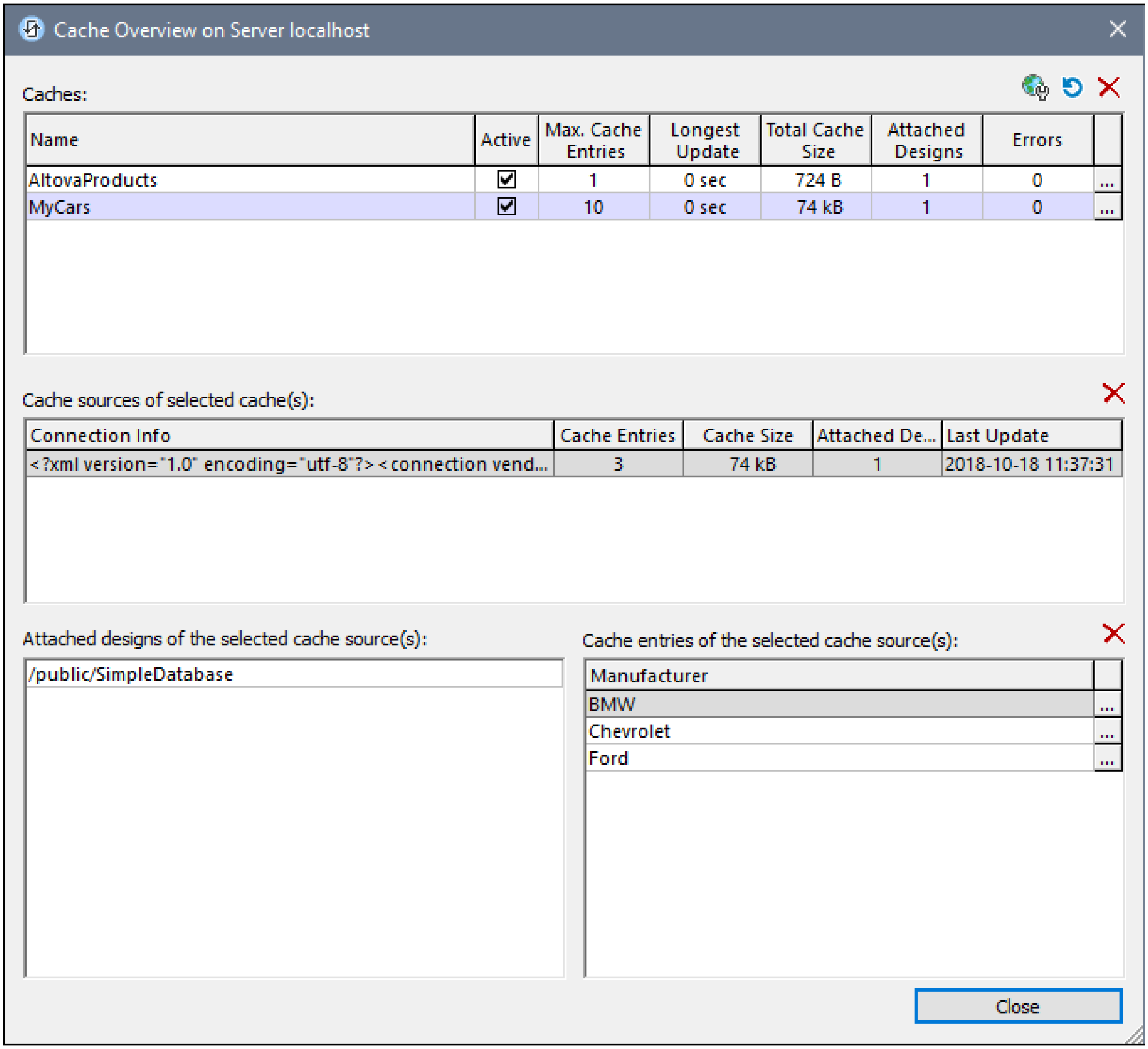
- *Name:* The name of the cache that is being created or edited.
- *Refresh button* (located on the right-hand side of the *Name* field)*:* Updates the connection to the server to check for the latest caches that match the structure of the current page source. If one or more caches have already been defined for the current page source, then the *Name* entry field becomes a combo box that enables you to select a cache from the available caches. If you wish to edit the selected cache, click **Change Cache Settings**. If you wish to create a new cache for the current page source, then click the **Add New Cache** button (located on the right-hand side of the *Name* field).
- *Add New Cache button* (located on the right-hand side of the *Name* field)*:* Adds a new cache for the current page source and enters a default cache name in the *Name* field. You can edit the name.
- *Change Cache Settings:* Enabled when an existing cache is selected in the *Name* field. Click to load the selected cache's settings and edit these. Click **Save** (at the bottom of the dialog) to save the modified settings.
- *Clear Cache:* Clear the cache on the server.
- *Active:* Activate/deactivate the cache.
- *Show an error:* Specify whether an error should be displayed if the cache is missing or is deactivated when the solution is run.
- *Save data in cache on first request:* Data is saved from the data source when the solution first requests the data.
- *Update cache periodically:* Select to specify the frequency with which the cache should be filled.
- *Save:* Saves the cache and its properties.
- *Save and Fill Cache:* Saves the cache and fills it with data from the data source

*Maximum cache entries*

If the page source that is being cached is a database that has a SELECT statement containing one or more parameters, then you can set a maximum number of cache entries. This number specifies how many cache entries will be stored before the first cache entry is deleted and the latest cache entry is appended. This setting: (i) enables a new cache entry to be created each time a new parameter value is found, and (ii) limits the amount of server space that will be used for caches of this particular page source. In order to judge the amount of space that will be used, you can check the size of the cache in the Cache Overview dialog (*see below*) or in the Cache Overview tab of MobileTogether Server.

## 7.4.2 Cache Overview

The Cache Overview dialog (*screenshot below*) is accessed with the menu command **Project | Cache Overview**. The Cache tab of MobileTogether Server also provides a cache overview and the same features as the Cache Overview dialog described here.



Caches:

| Name | Active | Max. Cache Entries | Longest Update | Total Cache Size | Attached Designs | Errors | |
|---|---|---|---|---|---|---|---|
| AltovaProducts | ☑ | 1 | 0 sec | 724 B | 1 | 0 | ... |
| MyCars | ☑ | 10 | 0 sec | 74 kB | 1 | 0 | ... |

Cache sources of selected cache(s):

| Connection Info | Cache Entries | Cache Size | Attached De... | Last Update |
|---|---|---|---|---|
| <?xml version="1.0" encoding="utf-8"?><connection vend... | 3 | 74 kB | 1 | 2018-10-18 11:37:31 |

Attached designs of the selected cache source(s):

/public/SimpleDatabase

Cache entries of the selected cache source(s):

| Manufacturer | |
|---|---|
| BMW | ... |
| Chevrolet | ... |
| Ford | ... |

The dialog provides, in the top pane, an overview of all the caches on the server. In the dialog, you can do the following:

- Activate/deactivate a cache (in the top pane).
- Delete a cache (in the top pane) by selecting it and clicking **Delete**.
- Select a cache in the top pane to display its details in the other (subsidiary) panes of the dialog.
- Delete a cache item in a subsidiary pane by selecting it and clicking the subsidiary pane's **Delete** button.
- Reconfigure a cache by clicking the **Additional Dialog** button (at the right-hand side of the cache entry in the top pane), and editing the cache's configuration.
- View a log of cache entry updates by clicking the **Additional Dialog** button in the Cache Entries pane (located at the bottom right of the dialog). In the screenshot above the `MyCars` cache has three cache entries, which are listed in the Cache Entries pane.

# 8 Pages and Page Events

A MobileTogether project consists of one or more pages, which are defined in an ordered sequence. When the MobileTogether application is started on the client device, the workflow starts at the first page of the sequence and progresses through the sequence to the last page. Top-level pages of the workflow sequence can branch off to subpages and then return to the main workflow. A project containing multiple pages thus enables you to structure a complex workflow into parts that are more comprehensible to the user. The pages of a project are managed in the Pages Pane 257. For an overview of pages, see the subsection Pages of a Project 381.

The design components of a page—typically the controls 403 that are added to the page—process data that comes from specified data sources. These data sources are defined in the Page Sources Pane 270 as page sources. After the page sources have been defined, they can be used by the controls and actions of that page. For an overview of how these "page sources" are to be used, see the subsection Data Sources of a Page 383.

There are a wide range of page properties that define various aspects of the page, such as its name and background color. These properties are listed and described in the subsection Page Properties 384.

An important feature of pages is that actions 667 can be set to execute when page-related events 389 are triggered. For example, every time a page is loaded, its page sources can be updated. The various page events and their settings are described in the subsection Page Events 389.

## In this section

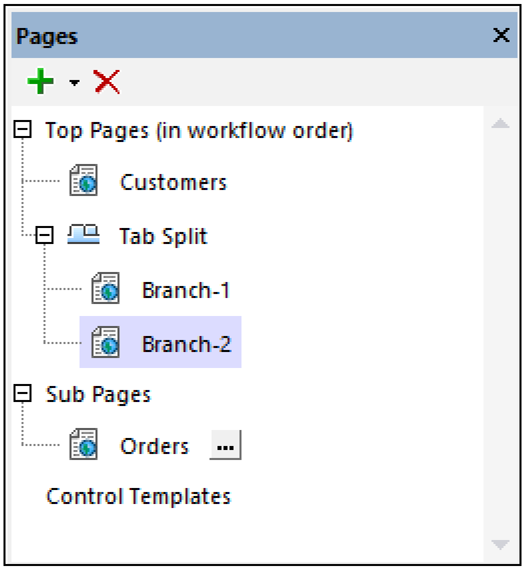
This section is organized into the following subsections:

# 8.1 Pages, Tabbed Pages, and Sub Pages

A MobileTogether project consists of one or more pages, which are defined in an ordered sequence. When the MobileTogether application is started on the client device, the workflow starts at the first page of the sequence and progresses through the sequence to the last page. Top-level pages of the workflow sequence can branch off to subpages and then return to the main workflow. A project containing multiple pages thus enables you to structure a complex workflow into parts that are more comprehensible to the user.

## Pages Pane: managing the pages of a project

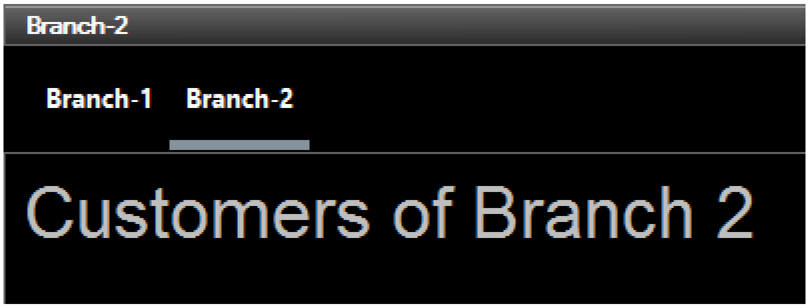
The pages of a project are managed in the Pages Pane 257 (*see screenshot below*). You can add three types of pages to your project: (i) top pages, (ii) tabbed pages (tab splits), and (iii) sub pages. Do this via the **Add** icon of the menu bar, or via the context menu of individual pages (right-click a page to see its context menu). The top-down order in which the pages are listed in the Pages Pane 257 is the page sequence of the project. When the solution is running in the client, clicking the **Submit** button will move the flow to the next page. You can change the page sequence of the project by dragging pages to new positions relative to the others.



For details about managing pages, see the description of the Pages Pane 257.

## Tabbed pages (tab split)

To add a tabbed page to the page sequence of a project, add a tab split to the sequence (*see screenshot above*). Then use the context menu of the tab split to add child pages to the tab split. For example, in the screenshot above, the tab split contains two child pages, the pages named *Branch-1* and *Branch-2*. The child pages of the tab split in the design will be the tabs of the tabbed page in the solution, as shown in the screenshot below. In the solution, the user can select a tab to see the corresponding page. When the user clicks the **Submit** button of the tabbed page, the workflow will move on to the next page in the page sequence. For more information about managing pages, see the description of the Pages Pane 257.



## Sub pages

Sub pages are added via the **Add** icon in the menu bar of the Pages Pane[257], or via the context menu of individual pages (right-click a page to see its context menu). You can move the flow from a top page to a sub page by using the Go to Subpage[783] action (for example, on a button control[417]). To go back to the top page, the user can click the MobileTogether Client app's **Back** button, Alternatively, you can define a Go to Page[783] action to go to any page in the design.

You can declare parameters for a sub page, as well as define variables that will be in scope in that sub page. The difference between parameters and variables is as follows:

- The values of parameters are defined externally and are evaluated when the sub page is called. In the Go to Subpage action[783], for example, when you select a sub page, the parameters of that sub page are displayed and you can define a value for each parameter. The sub page will then be passed the parameter values defined in the action.
- The values of variables, on the other hand, are defined on the sub page itself. This XPath expression can use the parameters of that control template as well as variables defined earlier in the list of variable definitions. For example: If a sub page has parameters **$a**, **$b**, **$c**, and variables **$x**, **$y**, **$z** (in that order), then the variable **$y** can use the following list of parameters and variables to generate its value: **$a**, **$b**, **$c**, **$x**, (but not **$z**).

**Note:** You can specify whether a parameter or variable is mandatory or optional and that it is an error if no value is passed to a mandatory parameter[783]. To edit define and edit the definitions of parameters and variables, click the **Add Parameters/Variables** button (which is located to the right of the sub page name) and define the parameters/variables in the Parameters dialog that appears.

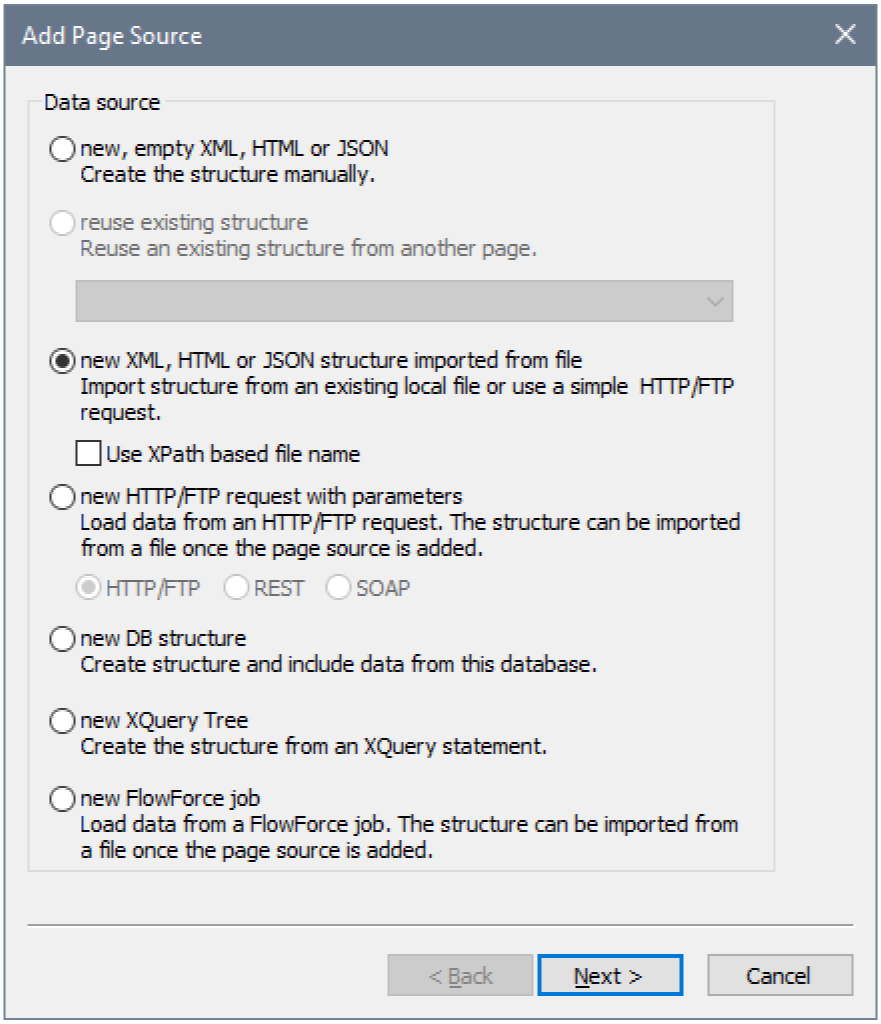
**Note:** You can change the order of parameters and variables by dragging a parameter/variable and dropping it to a new location.

**Note:** If you change the order of parameters, you need to make make sure that the parameter order is identical in the call to the subpage[783]. Do this as follows: (i) Right-click the sub page and select **List Usages**; (ii) In each usage (the list is shown in the Listings Pane[281]), check whether parameters are specified via an array or a list; (iii) If specified via a list, make sure that the order of parameters in the array is the same as that in the parameter definitions list (i the Parameters dialog).

For more information about managing pages, see the description of the Pages Pane[257].

## 8.2 Data Sources of a Page

Every page of a design has a set of page sources, which are managed in the Page Sources Pane[270]. To add a page source, click the pane's **Add Source** toolbar icon. This displays the Add Page Source dialog (*screenshot below*). Here you select the type of the page source[317] you want to add. If you want to add the same page source as one that you have used in an already existing page, then select the *Reuse existing structure* option and select from the existing page sources that are displayed in the combo box.



Add Page Source

Data source

○ new, empty XML, HTML or JSON
Create the structure manually.

○ reuse existing structure
Reuse an existing structure from another page.

● new XML, HTML or JSON structure imported from file
Import structure from an existing local file or use a simple HTTP/FTP request.

□ Use XPath based file name

○ new HTTP/FTP request with parameters
Load data from an HTTP/FTP request. The structure can be imported from a file once the page source is added.

● HTTP/FTP ○ REST ○ SOAP

○ new DB structure
Create structure and include data from this database.

○ new XQuery Tree
Create the structure from an XQuery statement.

○ new FlowForce job
Load data from a FlowForce job. The structure can be imported from a file once the page source is added.

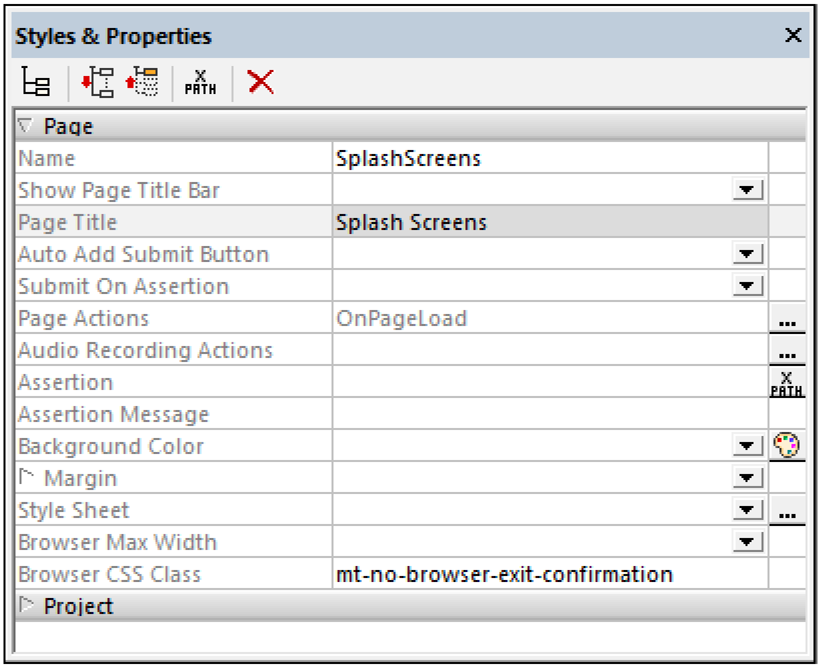
The second page of the Add Page Source dialog enables the following options to be set: (i) whether the page source is editable or not, when data is loaded from and saved to the page source, and whether data is stored on the client or server. For details of these page source options, see Page Source Options[345].

A number of settings for page sources and their individual nodes can be set separately. For a full listing of the context menu commands of different types of nodes, see Context Menus (of Page Sources)[359].

For more detailed information about page sources, see the sections Page Sources Pane[270] and Page Sources (Data Sources)[315].

## 8.3 Page Properties

Page properties are defined in the Styles & Properties Pane 274 and are described below. The screenshot below shows default values.



Styles & Properties

|  |  |
|---|---|
| ▽ Page |  |
| Name | SplashScreens |
| Show Page Title Bar |  |
| Page Title | Splash Screens |
| Auto Add Submit Button |  |
| Submit On Assertion |  |
| Page Actions | OnPageLoad |
| Audio Recording Actions |  |
| Assertion |  |
| Assertion Message |  |
| Background Color |  |
| Margin |  |
| Style Sheet |  |
| Browser Max Width |  |
| Browser CSS Class | mt-no-browser-exit-confirmation |
| Project |  |

### Name

The name of the page. It is used to reference the page within the project. If the **Page Title** property (*see below*) is not specified, it is also the title of the page in the solution. Click inside the value field and enter the name you want.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog 296 has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all page styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Background Color"  : $XML1/R/@background,
    "Margin   "         : "6dp"
   }

map{
    "Style Sheet"       : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the page in the Styles & Properties Pane 274 will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Background Color`** and **`Margin`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane 274.
- You can also specify a style sheet to use, as shown in the second map above.

### ▼ Show Page Title Bar

Specifies whether the title bar of the page is shown (`true`) or not (`false`). The default value is `true`. Note that this property refers to the presence or absence of the entire title bar. It does not refer to the content of the title bar, which is specified by the **`Page Title`** property.

### ▼ Page Title

The title of the page in the solution. Click inside the value field and enter the name you want. Alternatively, you can enter an XPath expression by clicking the **XPath** icon in the pane's toolbar. If a value for this property does not exist, then the value of the **`Name`** property will be used as the title of the page in the solution.

### ▼ Auto-Add Submit Button

A boolean setting that defines whether a **Submit** button is automatically added to the page. Select `true` or `false` in the combo box. The default value is `true`. (The **Submit** button of a page in the solution is usually located at the top right of the page, and it submits data on the page for action. Typically, the workflow then moves on to the next page.)

### ▼ Submit on Assertion

Allows or disallows a page submission if there are invalid assertions on the page. Select from the following values:

- `Disable`: The **Submit** button is disabled if there is an invalid assertion on the page. This is the default setting.
- `Enable`: The **Submit** button is enabled even if there is an invalid assertion on the page.
- `Ask`: The **Submit** button is enabled even if there is an invalid assertion on the page. However, if there is an invalid assertion, and the **Submit** button is clicked/tapped, then a dialog appears asking the end-user whether submission should proceed or not.

The default is `Disable`.

### ▼ Page Actions

Clicking the property's **Additional Dialog** button displays the Page Actions dialog 389, in which you can select a page event and then define actions to perform when a page event is triggered. See the sections

Page Events[389] and Actions[667] for details of how to do this. Page events for which actions have been defined are listed in the property's value field.

### Audio Recording Actions

Audio Recording events are defined per page. Two events are available: `OnAudioRecordingError` and `OnAudioRecordingFinished`. The actions that are defined for these events **apply to all Audio Recordings on the page**. Clicking the property's **Additional Dialog** button displays a dialog containing the definitions of the Audio Recording events of the current page. For each event, you can define the actions to perform by dragging and dropping actions from the left-hand Actions pane into the event's tab. The Audio Recording events dialog can also be accessed by right-clicking in the design and selecting **Page Audio Recording Actions**. *For more information, see the description of the Audio Recording feature*[1108].

### Assertion

Sets a condition to be met for the page to be valid. If the assertion is invalid, then the text of the `Assertion Message` property (*see next property below*) is displayed in the Assertion Message[409] control. (If there are multiple Assertion Message[409] controls, then all these controls will display the text of the `Assertion Message` property.)

Click the `Assertion` property's **XPath** icon to enter an XPath expression that defines the assertion. For example: The XPath expression `LastName != ""` asserts that the node `LastName` must not be empty. If this node is empty, then the assertion message of the page (defined in the `Assertion Message` property) is displayed in the page's Assertion Message[409] control.

Note that assertions can also be defined for some controls. So it can happen that there are multiple invalid assertions on a page. If this happens, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

### Assertion Message

Sets the assertion message to be displayed if the page assertion (*see previous property above*) is not valid. Double-click inside the value field of the property to edit the assertion message, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. The assertion message is displayed by the Assertion Message[409] control. For example: If the XPath expression of a page assertion is `LastName != ""`, then it asserts that the node `LastName` must not be empty. If this node is empty, then the assertion message of the page is displayed in the Assertion Message[409] control of the page.

Note that assertions can also be defined for some controls. So it can happen that there are multiple invalid assertions on a page. If this happens, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

### Background Color

Sets the background color of the object. You can do one of the following to select the color:

- Click the color palette to select a background color

- Select a color from the dropdown list of the combo box
- Double-click in the value field and enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the (color code) text you want

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* [1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** [1304] and **`$MT_CanvasY`** [1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the project [1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* [1327]). See the section Style Sheets [1318] for more information.

▼ Browser Max Width

Specifies the maximum page width in browsers. Select a value from the dropdown list, or click the window's **XPath** toolbar icon to specify the width via an XPath expression. Values can be given as (i) a percentage of the browser width, (ii) a dp value (density-independent pixels), (iii) an sp value (scale-independent pixels), or (iv) an absolute pixel value. The default value is the available browser width. Note that the page width will be limited to the maximum you specify only if the screen width is larger than the specified value. *See Sizes: Pixels, DPI, DP, SP*[1312] *for more information.*

▼ Browser CSS Class

Enter the name of the CSS class that you want to associate with this page. This class can then be used in a CSS file (specified in the Project Properties[296]) to assign properties for this control separately.

The following values set specific predefined behavior:

- **`mt-no-browser-exit-confirmation`** disables the page-exit confirmation dialog from appearing for **this page**—if the page-exit confirmation option has been set in the Browser Settings of the project[296] (that is, for all pages).
- **`mt-combo-open-on-focus`** opens the dropdown list of any combo box on the page when the user tabs to it. Since this value is set on the `Browser CSS Class` property of the page, it applies to all combo boxes on the page. Alternatively, you can restrict the scope (that is, which combo boxes are opened on focus) by setting the this property value on the `Browser CSS Class` property of a single combo box[459] or of a table containing combo boxes[614].

# 8.4 Page Events

Each page in a MobileTogether design has certain predefined **events** associated with it:

- OnPageLoad 390: You can define a set of actions to execute when the page is loaded.
- OnPageRefresh 390: You can define when a page is refreshed: (i) when the page is reopened; (ii) at timer intervals; (iii) manually, when the user taps/clicks the **Refresh** button, or (on iOS) pulls down to refresh. Actions can be defined for each of these methods, and one or all can be used.
- OnBackButtonClicked 393: Actions to execute when the **Back** button of the solution is tapped/clicked.
- OnSubmitButtonClicked 394: Actions to execute when the **Submit** button of the solution is tapped/clicked.
- OnServerConnectionError 394: Actions to execute when the server cannot be reached. The error could occur while making the initial connection, or, subsequently, when the connection is lost. Use the MT_ServerConnectionErrorLocation 1304 variable to debug such errors. For an overview of how this event can be used, see the section Server Connection Errors 394.
- OnEmbeddedMessage 397: An event that generates the message that is posted from an IFrame embedded in a webpage to the workflow on the server. See Embedded Webpage Solutions 1427 for a description of this feature.
- OnMQTTReceive 399: This event is triggered when an MQTT message is received for any MQTT subscription defined on the page.
- OnProgressUpdate 401: An event on subpages that indicate the progress of server actions. It is triggered by the Progress Update 796 action and can update the subpage with information about server-action progress.

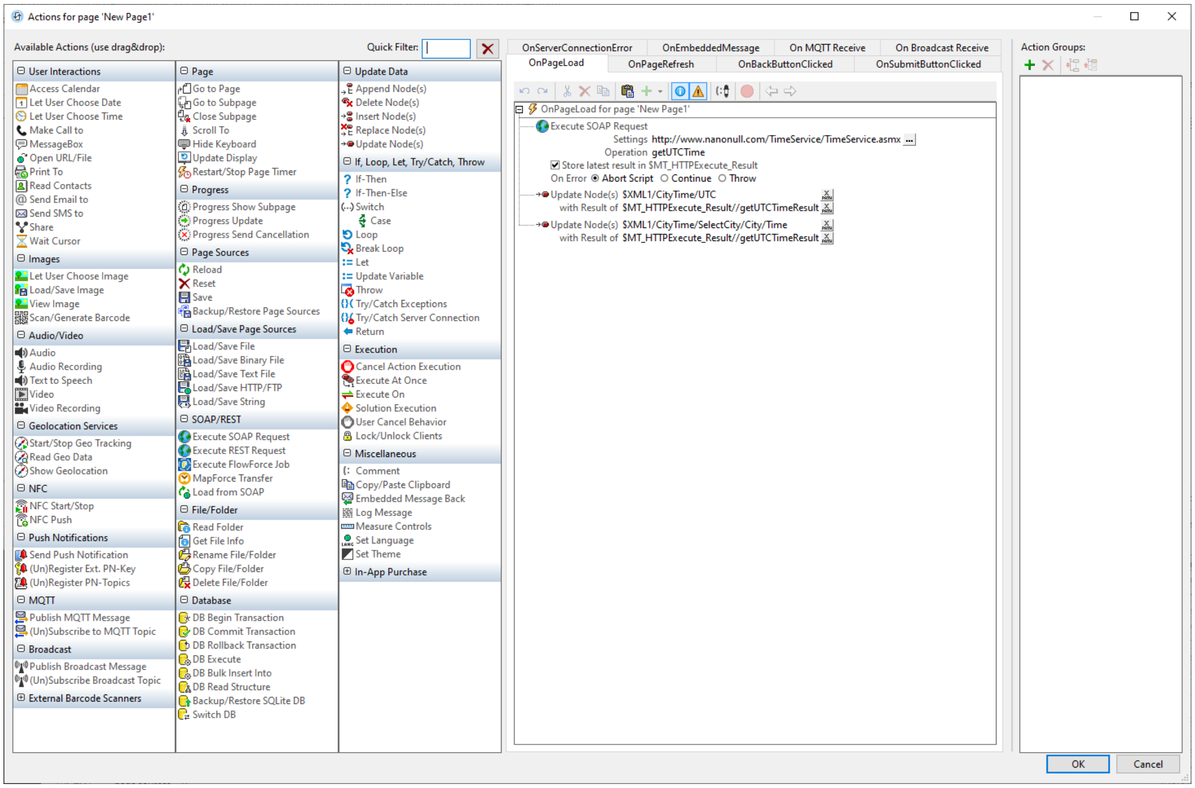
These events are called **page events** (as opposed to control events 665), and each can have one or more **actions (page actions)** 667. When a page event is triggered, the page action that is defined for it is carried out.

## Defining the actions of a page event

To define the actions of a page event, access the Page Actions dialog 667 (*screenshot below*). Select the tab of the event you want, then define its actions.

You can access the Page Actions dialog 667 (*screenshot below*) in one of the following ways:

- Click **Page | Page Actions** 1632
- Right-click anywhere in the page, and select **Page Actions**
- In the Styles & Properties Pane 274, go to the *Page* section. and click the **Additional Dialog** button of the `Page Actions` property
- Click **Page | Actions Overview** 1632 to display the Actions Overview dialog 1632. Then click the **Additional Dialog** button of the page event you want to define



OnPageLoad for page 'New Page1'

Execute SOAP Request

Settings http://www.nanonull.com/TimeService/TimeService.asmx

Operation getUTCTime

☑ Store latest result in $MT_HTTPExecute_Result

On Error ● Abort Script ○ Continue ○ Throw

Update Node(s) $XML1/CityTime/UTC
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

Update Node(s) $XML1/CityTime/SelectCity/City/Time
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

For information about the Actions dialog, the dialog's panes, and pane features, see the section Actions 667.

## Defining page actions

To define the action/s of a page event, drag the desired action from the left-hand pane into the event tab in the right-hand pane. For more information about the Page Actions dialog 667 and on the different types of available page actions, see the section Actions 667.

## 8.4.1 OnPageLoad

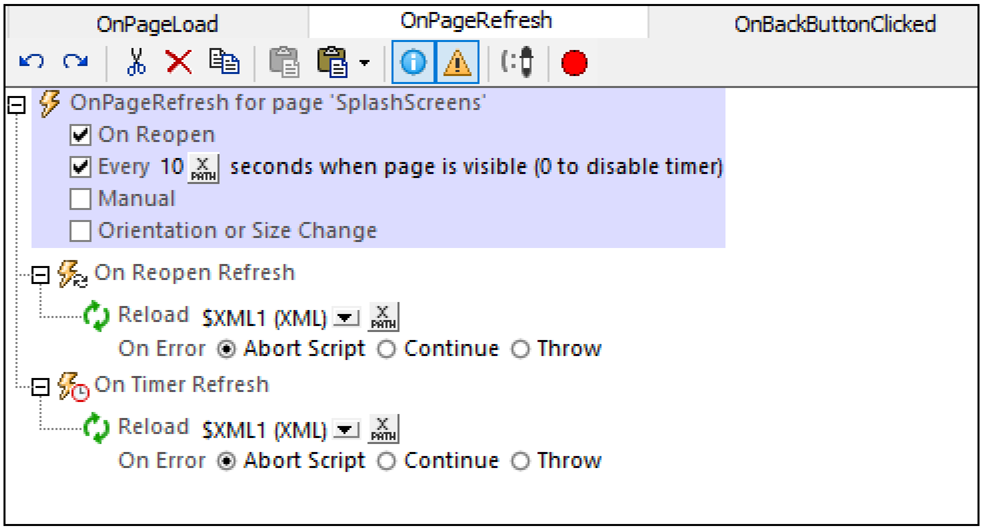
The `OnPageLoad` event is available for all pages (top pages and sub pages 257). Actions that are defined for the `OnPageLoad` event are executed when the page loads. These actions can be useful for initializing data values and setting up components in the display.

## 8.4.2 OnPageRefresh

The **`OnPageRefresh`** event is available for all pages (top pages and sub pages 257). It can be defined to occur in one or more of the following cases (*see screenshot below*).

- Whenever the page is reopened (or resumed from a paused state).

- At intervals determined by a timer that is started when the page is first loaded. The interval can be a static or dynamic value, and the timer can be stopped and restarted when other (page or control) events are triggered.
- Manually, when the user taps/clicks the **Refresh** button at the top of the page, or (on iOS) pulls down to refresh.
- Whenever the orientation (portrait or landscape) is changed, or the app window is re-sized.



OnPageRefresh for page 'SplashScreens'
☑ On Reopen
☑ Every 10 seconds when page is visible (0 to disable timer)
□ Manual
□ Orientation or Size Change

On Reopen Refresh
Reload $XML1 (XML)
On Error ● Abort Script ○ Continue ○ Throw

On Timer Refresh
Reload $XML1 (XML)
On Error ● Abort Script ○ Continue ○ Throw

When you select an option, a node for that option appears in the tree (*On Reopen Refresh, On Timer Refresh, On Manual Refresh*). You can define actions for one or more of these options on their respective nodes. In the screenshot above, the page has been set to refresh in two situations: (i) every time the page is reopened, **and** (ii) every 10 seconds. In both cases, the same Reload action has been defined. You can specify the same or different action/s for each node.

*On Reopen Refresh*

The actions defined for this option are executed when a page is reopened, and also when a solution that was paused (and is running in the background) is reopened. Also see the project property *On Switch to Other Solution*[296] and the `SolutionExecution`[915] action.

*On Timer Refresh*

- The timer interval is selected with an XPath expression. The value must be a number; it is read as the refresh interval in seconds. (The allowed precision is in the milliseconds. So a value of **`1.002`** is allowed, and sets a refresh interval of 1 second and 2 milliseconds.) The default is a static value of **`10`** (seconds). You can also set a dynamic value (for example, a number value from a page source node, or a value that is generated by a calculation).
- The timer is first started when the page is loaded. The *On Timer Refresh* actions are executed at intervals determined from this start time onwards. If you change the refresh interval, then the timer must be restarted. This is done by adding the Restart Page Timer[791] action to the event that changes the refresh interval. *See the SOAP Requests tutorial[226] for an example.*
- The refresh actions will continue to be executed at the specified intervals as long as the timer runs. To stop the timer, add the Stop Page Timer[791] action to a suitable event.

*On Manual Refresh*
If this option is selected, the page will display a **Refresh** button. When the user taps/clicks this button (or, on iOS, pulls down to refresh), the *On Manual Refresh* actions are executed. *See the SOAP Requests tutorial*[226] *for an example*.

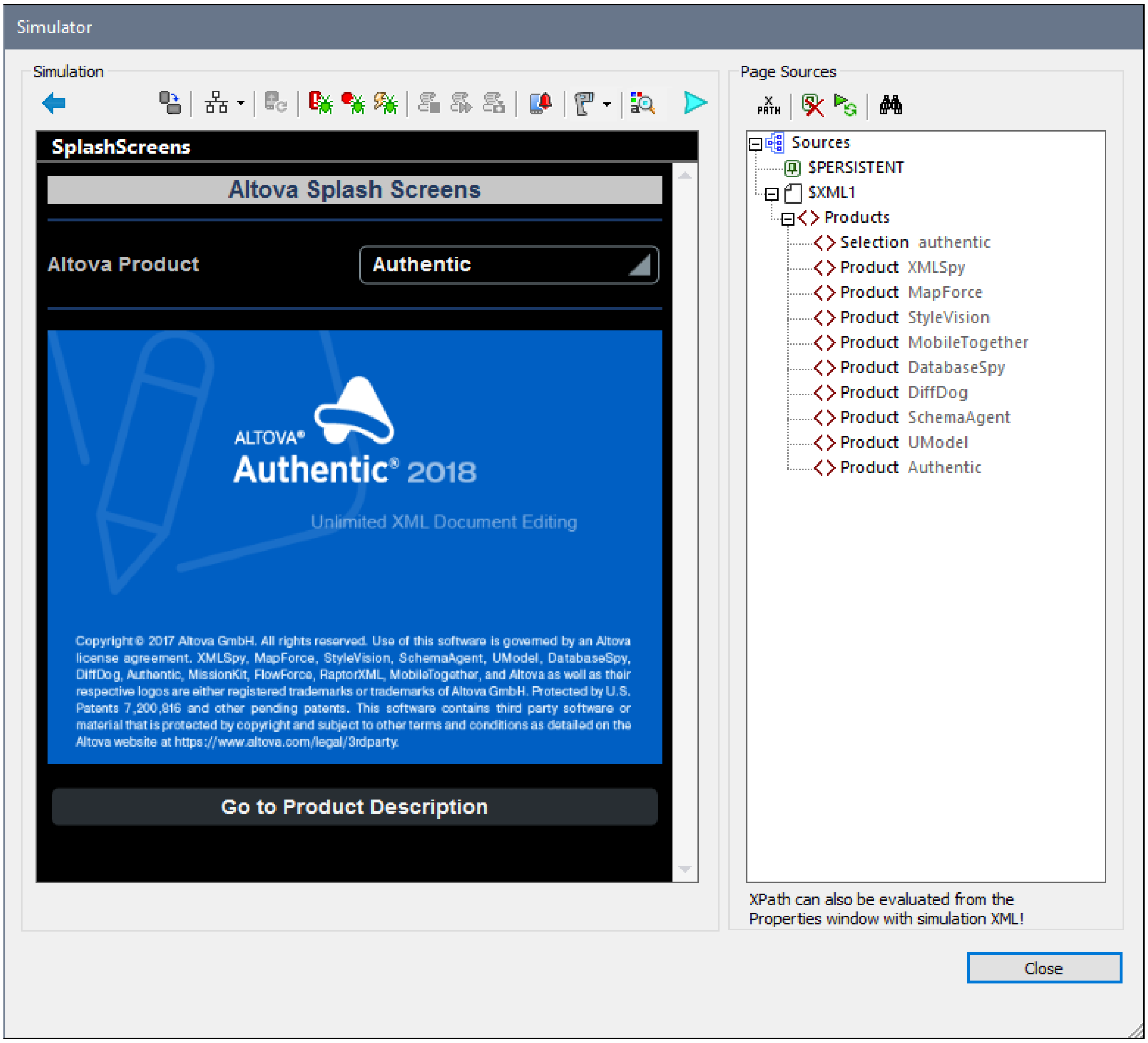
*Refresh due to orientation change or resizing of app window*
The actions defined for this option are executed when the end user changes device orientation (between portrait and landscape) or re-sizes the app window (on devices where the window is re-sizable). For example, if the device orientation is changed from landscape to portrait, you can reduce the number of columns in a table. The following variables are particularly useful for the actions defined here: device-dimension variables[1300], device-orientation variables[1304], variables for device viewport dimensions[1304] and variables for window size[1304].

For an example of page refreshes, see the *SOAP Requests tutorial*[226].

## Simulating page refreshes

In the Simulator (*screenshot below*), you can influence page refreshes in the following ways:

- If a page refresh is defined on page reopens, then the **Simulate Reopen** button is enabled. Click it to simulate a page reopen.
- If a time-based page refresh is defined, then the **Start/Stop Timers** button is enabled. When the simulation starts, the page is automatically refreshed every X seconds, where X is the refresh interval. You can stop the refreshes by clicking **Stop Timers**. This is useful if you wish to look at the progress of the simulation without having the page being constantly refreshed. When the timer is stopped, the button changes into a **Start Timers** button, which you can click to re-start the timer.
- If a manual refresh is defined, then a **Refresh** button is available. Click it to execute the actions specified for the *On Manual Refresh* option.



Copyright © 2017 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,200,816 and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at https://www.altova.com/legal/3rdparty.

```
Sources
  $PERSISTENT
  $XML1
    Products
      Selection  authentic
      Product  XMLSpy
      Product  MapForce
      Product  StyleVision
      Product  MobileTogether
      Product  DatabaseSpy
      Product  DiffDog
      Product  SchemaAgent
      Product  UModel
      Product  Authentic
```

XPath can also be evaluated from the Properties window with simulation XML!

## 8.4.3 OnBackButtonClicked

The `OnBackButtonClicked` event is available for all pages (top pages and sub pages[257]). In the solution display and in simulations, the **Back** button is located at the top left of the screen.

The default behavior of the **Back** button depends on the page type[257]:

- *Top pages:* A prompt appears asking whether you wish to exit the solution. Click **Yes** to exit, **No** to cancel.
- *Sub pages:* The display switches to the previous page, which would typically be the top-level page that loaded the sub page.

Defining a set of actions for this event overrides the default behavior listed above. Only the defined event actions are executed when the user taps/clicks the **Back** button.

## 8.4.4 OnSubmitButtonClicked

The **Submit** button appears at the top right of all Top Pages 257—if the Auto-Add Submit Button 384 property of the top page has been left at its default value of **`true`** in the Page Properties 384 settings.

The default behavior of the **Submit** button depends on the position of the page in the page sequence.

- If the page is the last page in the page sequence, tapping/clicking the **Submit** button exits the workflow.
- If the page is not the last page, then the workflow goes to the next page.

If you wish to specify any other action when the **Submit** button of a page is tapped/clicked, then add the appropriate actions to the `OnSubmitButtonClicked` event of the page.

## 8.4.5 OnServerConnectionError

The `OnServerConnnectionError` event is available for all pages (top pages and sub pages 257). A server connection error could occur while making the initial connection, or when the connection is subsequently lost. For the event that such an error occurs, you can define:

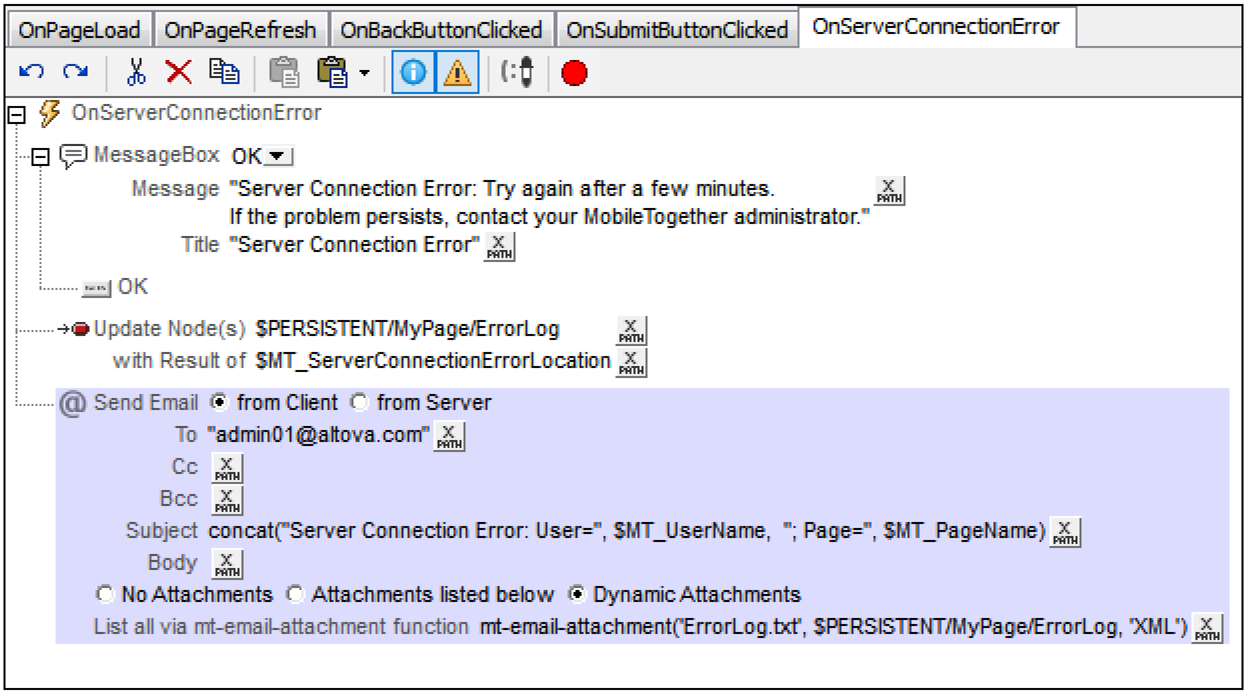
- a suitable error message to the end user (client device), and
- the subsequent actions to take.

You can also simulate a connection error in the Simulator 1355.

**Note:** When a server connection error occurs, the first of the following actions that exists is triggered: (i) a Try/Catch Server Connection Errors 908 action, (ii) action/s for the **`OnServerConnectionError`** 394 event (this event), (iii) a MobileTogether message about the error, following which the workflow is resumed.

### Defining what to do when there is a connection error

You can define what actions to execute when there is a server connection error. These actions are defined for each page: in the tab of the page event 389 `OnServerConnectionError`. The actions you define would typically include a message to the end user and a procedure for the workflow to follow. The screenshot below shows a sequence of actions that could be performed.



OnServerConnectionError

MessageBox OK

Message "Server Connection Error: Try again after a few minutes.
If the problem persists, contact your MobileTogether administrator."

Title "Server Connection Error"

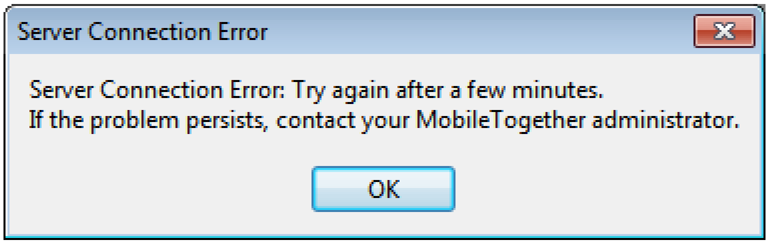
OK

Update Node(s) $PERSISTENT/MyPage/ErrorLog
with Result of $MT_ServerConnectionErrorLocation

Send Email ● from Client ○ from Server
To "admin01@altova.com"
Cc
Bcc
Subject concat("Server Connection Error: User=", $MT_UserName, "; Page=", $MT_PageName)
Body
○ No Attachments ○ Attachments listed below ● Dynamic Attachments
List all via mt-email-attachment function mt-email-attachment('ErrorLog.txt', $PERSISTENT/MyPage/ErrorLog, 'XML')

The screenshot above defines a sequence of three actions to perform:

1. Send an error message to the client (*screenshot below*).

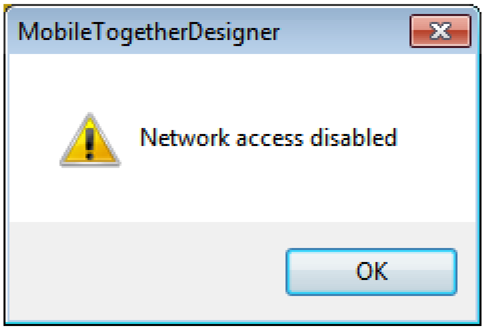


Server Connection Error

Server Connection Error: Try again after a few minutes.
If the problem persists, contact your MobileTogether administrator.

2. Use the `MT_ServerConnectionErrorLocation` 1304 variable to save the action stack that triggered the `OnServerConnectionError` 389 page event. (The variable should be used for debugging purposes; see `MT_ServerConnectionErrorLocation` 1304 for details.) Alternatively to the `MT_ServerConnectionErrorLocation` variable, you can use the `Update Node` 886 action to write your own error codes into a node that you specially create for this purpose.
3. Send an email to the administrator (from the client) with the error information as an attachment.

**Note:** If no action is defined in the tab of the `OnServerConnectionError` page event 389, then a generic *Network access disabled* message is sent to the mobile device (*screenshot below*):

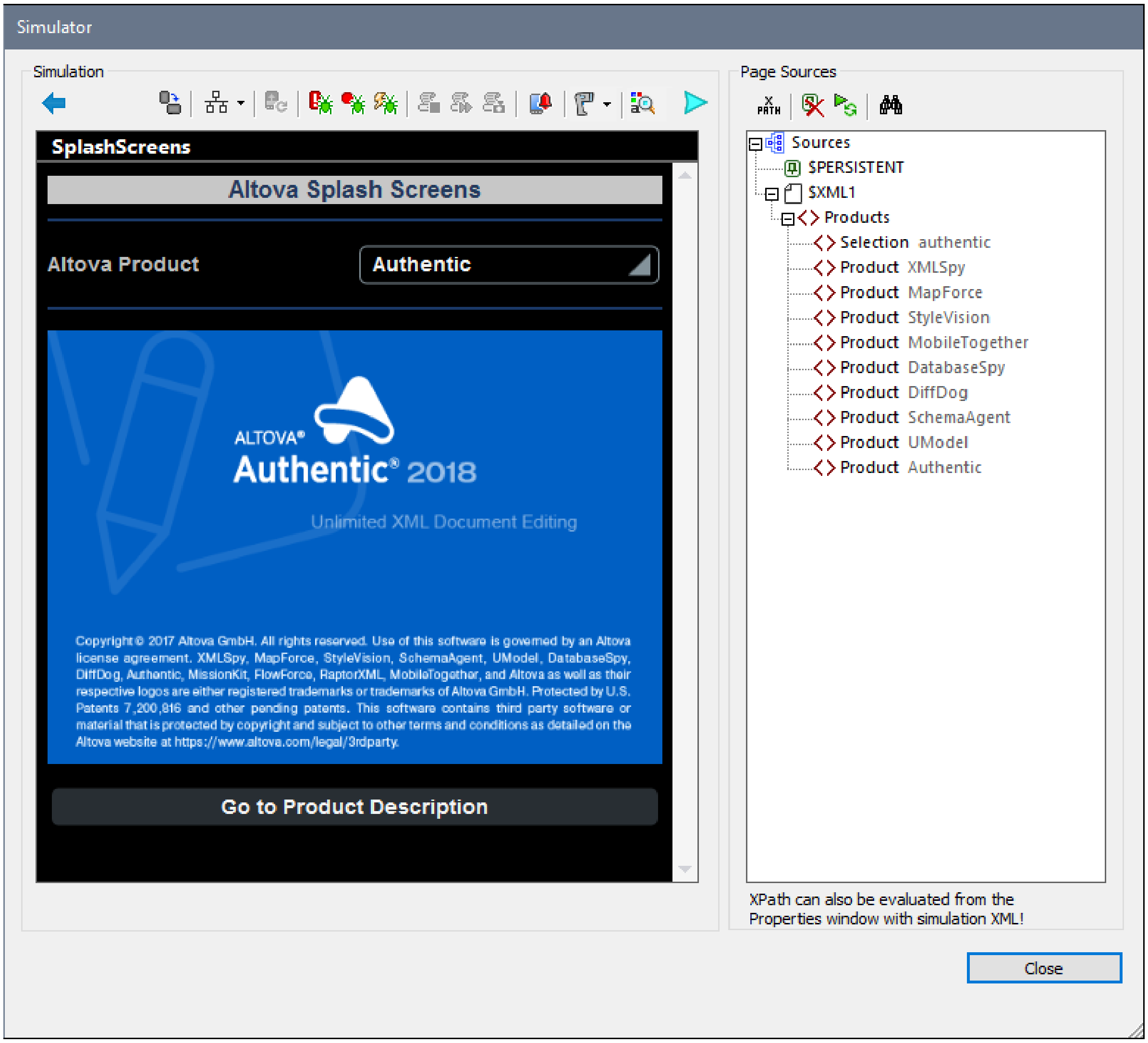


Network access disabled

## Simulating a server connection error (for testing)

You can use simulations in MobileTogether Designer[1356] and simulations on the Server[1362] to test the behavior of a solution. Do this as follows:

1. Start the simulation (for example, with **F5**). The simulator starts (*screenshot below*).



Copyright © 2017 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,200,816 and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at https://www.altova.com/legal/3rdparty.

XPath can also be evaluated from the Properties window with simulation XML!

2. Click **Prevent Server Access**. Server access will be disabled, and the button will toggle into an **Enable Server Access** button.
3. Carry out an action that calls for a server connection. Since access is disabled, the actions defined in the `OnServerConnectionError` 389 page event are triggered.
4. To enable server access again, click **Enable Server Access** in the simulator.

## 8.4.6 OnEmbeddedMessage

The `OnEmbeddedMessage` event is available for all pages (top pages and sub pages 257). An **embedded message** is a message that is posted from an IFrame embedded in the webpage to the workflow on the server. The workflow expects to receive a serialized JSON string.

The `OnEmbeddedMessage` event works as follows:

- At *design time:* When an action is defined for the OnEmbeddedMessage event, a JSON page source called $MT_EMBEDDEDMESSAGE is created with a root element named json. Additional nodes can be manually added to the page source. Design components can then access the nodes of this page source using XPath expressions. **Note:** Activating the OnEmbeddedMessage event (by adding an action to it) is one of two ways to create the **$MT_EMBEDDEDMESSAGE** page source at design time. (The other way is to define an Embedded Message Back[924] action anywhere on the page.)
- At *run time:* The OnEmbeddedMessage event creates a JSON page source called **$MT_EMBEDDEDMESSAGE** with a root element named **json**. The structure and content of this page source will come from data in the embedded message. If this page source does not have the same structure as that of the page source created at design time, then the run-time page source cannot be accessed by the XPath expressions of design components.

**Note:** The OnEmbeddedMessage event is a page event, so each page in the design can have actions defined for its OnEmbeddedMessage event. When a message is sent from the webpage, it is sent to the workflow as a whole (without specifying any particular page). It is the OnEmbeddedMessage event of the currently active page that will be triggered.

## At design time

The **$MT_EMBEDDEDMESSAGE** JSON page source is automatically created if at least one action is defined for the event handling of OnEmbeddedMessage. If you wish to carry out no other action except to create the page source, then add an action that does not interfere with the workflow, for example, a Comment[923] action. If you do not add an action, then the **$MT_EMBEDDEDMESSAGE** page source will not be created. If you subsequently remove all the defined actions, then the **$MT_EMBEDDEDMESSAGE** page source will also be removed.

When the **$MT_EMBEDDEDMESSAGE** page source is created, it will have a root element named **json**—and nothing else. You can manually add a JSON structure to the page source by using the commands in the toolbar of the Page Sources Pane[270]. A structure is required because it enables design components to access these nodes via XPath expressions.

The design-time and run-time structures of the **$MT_EMBEDDEDMESSAGE** page source must match. Otherwise, XPath expressions in design components might not be able to locate run-time page source nodes.

## At run time

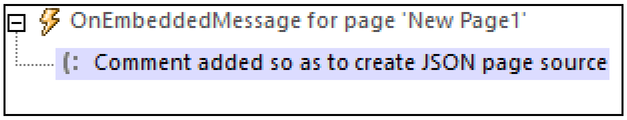
At run time, if an embedded message is received in the form of a JSON string, then the OnEmbeddedMessage event creates the **$MT_EMBEDDEDMESSAGE** page source. (Otherwise, it does not create this page source.) The page source is created with a root element named **json**, and will have the structure and data contained in the message. If additional actions have been defined for the event, then these are executed.

## Using the embedded message in the solution

The OnEmbeddedMessage event expects the message it receives to be a JSON string, and it parses this string to generate the **$MT_EMBEDDEDMESSAGE** page source, which is a JSON page source.

Two scenarios arise depending on the format of the source data in the webpage:

- *JSON format:* The JSON page source **$MT_EMBEDDEDMESSAGE** can be used directly in the solution, and data from this page source can be passed back to the webpage. In this case, any non-intrusive action (such as the Comment[923] action) can be defined for OnEmbeddedMessage (*see screenshot below*). This activates event handling and suffices to create the **$MT_EMBEDDEDMESSAGE** page source.

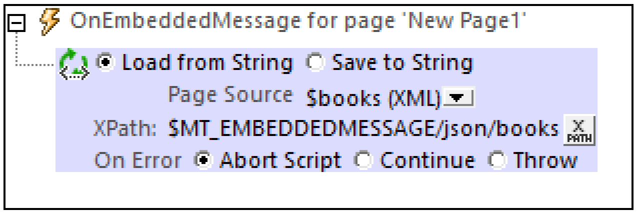


⊟ OnEmbeddedMessage for page 'New Page1'

(: Comment added so as to create JSON page source

*See [Sending/Receiving JSON Data](1445) for an example of how to work with JSON data.*

- *XML format:* In addition to the JSON page source **`$MT_EMBEDDEDMESSAGE`**, which is automatically generated by `OnEmbeddedMessage`, an XML page source should also be created. This will enable the solution to send data in a form that can more easily be converted back to the XML format of the webpage source. In order to create an XML page source from the data in **`$MT_EMBEDDEDMESSAGE`**, use the [Load from String](829) action (*see screenshot below*). In order for this to work, the selected node in **`$MT_EMBEDDEDMESSAGE`** must contain a string that can be parsed as XML.



⊟ OnEmbeddedMessage for page 'New Page1'

● Load from String ○ Save to String

Page Source $books (XML)

XPath: $MT_EMBEDDEDMESSAGE/json/books

On Error ● Abort Script ○ Continue ○ Throw

*See [Sending/Receiving XML Data](1453) for an example of how to work with XML data.*

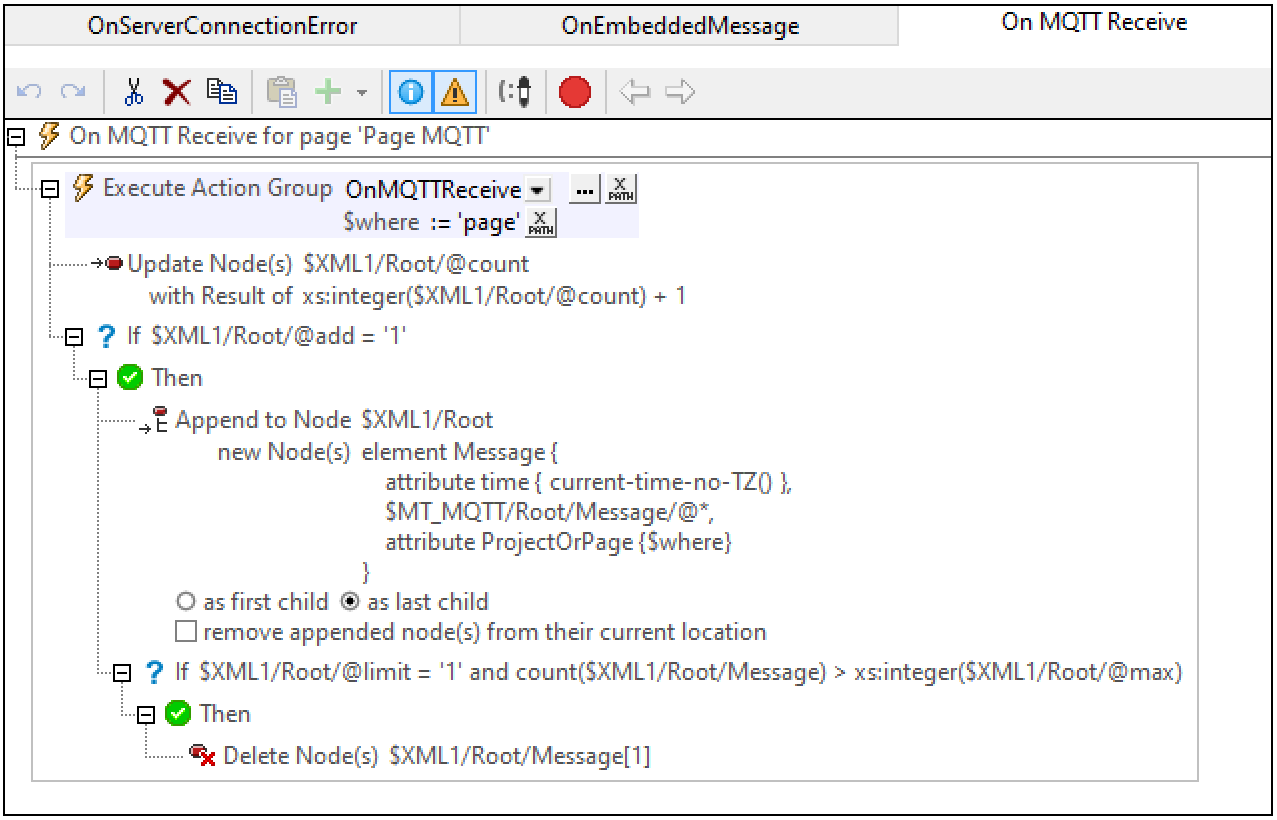
## 8.4.7 OnMQTTReceive

The **`OnMQTTReceive`** page event is available for all pages ([top pages and sub pages](257)). It is triggered when the solution receives a message because of a [Subscribe to MQTT Topic](764) action that has been defined on the page. When a message is received, the contents of the message are stored in the [`$MT_MQTT`](1138) page source and the actions defined here, on the **`OnMQTTReceive`** page event, are executed.

### Actions to perform when an MQTT message is received

You can define what actions to execute when an MQTT message is received. Do this by dragging and dropping actions into the event's main pane (*screenshot below*).

Note that the information contained in the message will automatically be passed to the [`$MT_MQTT`](1138) page source. This information will comprise two pieces of data: (i) the MQTT message, which will be stored as a text string, (ii) the name of the topic under which that message was sent. Now you could, for example, add the new message to an XML file that contains all the messages that were received for topics on that page. The screenshot below shows a sequence of actions does this, by appending a new **`Message`** node to the **`$XML1`** page source and deleting any older message if a limit of one message has been set for the **`$XML1`** page source.



```
⊟ On MQTT Receive for page 'Page MQTT'
  ⊟ Execute Action Group  OnMQTTReceive
                          $where  := 'page'
      Update Node(s)  $XML1/Root/@count
        with Result of  xs:integer($XML1/Root/@count) + 1
    ⊟ ? If  $XML1/Root/@add = '1'
      ⊟ Then
          Append to Node $XML1/Root
              new Node(s)  element Message {
                              attribute time { current-time-no-TZ() },
                              $MT_MQTT/Root/Message/@*,
                              attribute ProjectOrPage {$where}
                           }
          ○ as first child  ● as last child
          □ remove appended node(s) from their current location
        ⊟ ? If  $XML1/Root/@limit = '1' and count($XML1/Root/Message) > xs:integer($XML1/Root/@max)
          ⊟ Then
              Delete Node(s)  $XML1/Root/Message[1]
```

Note the following points:

- The actions defined on this event will be triggered each time a message is received for any and all subscriptions defined on the page.
- The data currently in the `$MT_MQTT` 1138 page source will contain the data of the message that triggers the page event.
- If you want the MQTT actions that you set on this page to be the same as those on another page, then consider defining these actions once at the solution (or project) level. MQTT actions at the project level 296 will be triggered if no **`OnMQTTReceive`** action has been defined at the page level. The actions at the project level, therefore, act as a fallback set of actions.

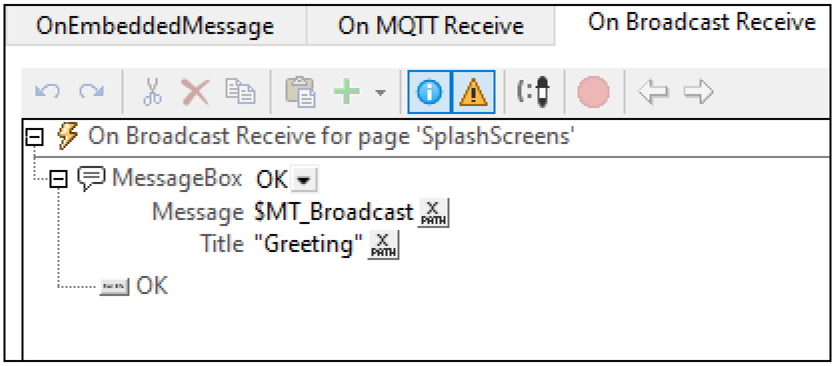
## 8.4.8 OnBroadcastReceive

The **`OnBroadcastReceive`** page event is available for all pages (top pages and sub pages 257). It is triggered when the solution receives a message because of a Subscribe to Broadcast Topic 764 action that has been defined on the page. It can make use of the **`$MT_Broadcast`** 1304 variable, which contains the text of the received broadcast message.

### Actions to perform when a broadcast message is received

You can define what actions to execute when a broadcast message is received. Do this by dragging and dropping actions into the event's main pane (*screenshot below*).

In the screenshot below, for example, our action tree displays a message box when the broadcast message is received. The title of the message box has been set to *Greeting*, and the text contained in the message box is the text of the received broadcast message (which has automatically been stored in the **`$MT_Broadcast`** 1142 variable.



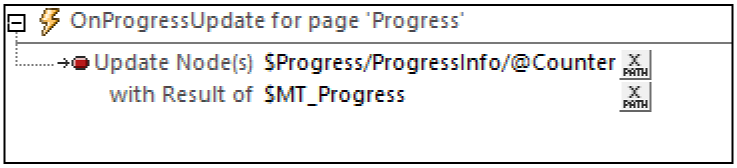
On Broadcast Receive for page 'SplashScreens'
MessageBox OK
Message $MT_Broadcast
Title "Greeting"
OK

Note the following points:

- The actions defined on this event will be triggered each time a broadcast message is received for any and all broadcast subscriptions defined on the page.
- The text of the received broadcast message is stored in the **`$MT_Broadcast`** 1142 variable and can be used in the action tree of the page event.
- The content of **`$MT_Broadcast`** 1304 will not be available after the action tree has finished executing.
- If you want to use the text of the received broadcast message after the action tree has finished executing, make sure that one of the actions passes the value of **`$MT_Broadcast`** 1304 to a page source node.
- If no actions have been set on this page event, then the **`OnBroadcastReceive`** actions at the project level will be executed—if defined. If no actions are defined at the project level either, then no action will be performed when a broadcast message is received.

## 8.4.9 OnProgressUpdate

The `OnProgressUpdate` event (*see screenshot below*) is available on sub pages 257 only. This is because the event is relevant only for those subpages that report progress of server actions.



OnProgressUpdate for page 'Progress'
Update Node(s) $Progress/ProgressInfo/@Counter
with Result of $MT_Progress

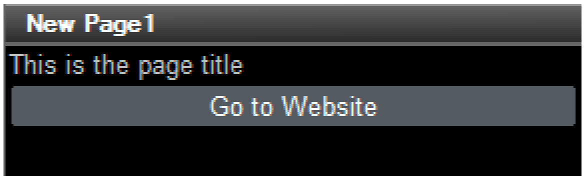
The `OnProgressUpdate` event is triggered when the progress of server actions has moved forward another step as defined by the Progress Update 796 action. This action updates, via its *Value* property, the **`$MT_Progress`** 1304 variable. You can now use the variable in the actions of the `OnProgressUpdate` event. For example, in the screenshot above, a node in one of the page sources of the *Progress* subpage has been updated with the value contained in **`$MT_Progress`** 1304. Since this page source node would be updated continually—in fact, each time the Progress Update[796] action is executed—it will contain a continuous record of the server-action progress, and you can display this progress information on the subpage.

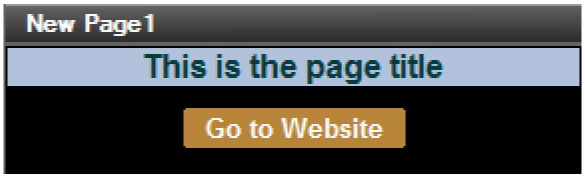
See the Progress Indicator tutorial[241] for an example of how the `OnProgressUpdate` event is used.

# 9 Controls and Control Events

The controls that appear in a page determine the design of the page, both in terms of **functionality** as well as of **appearance**.

For example, if you wish to make a simple page that consists of (i) a page title, and (ii) a button that links to a website (*see screenshots below*), then you would add two controls to the page: (i) a label control[554] that displays the page title, and (ii) a button control[417] that links to the website. The website page is opened when the end user clicks the button. The click triggers a button action that links to the website. In this way, the button control[417] provides **functionality**. At the same time, the two controls can be styled by specifying a wide range of properties for the two controls (label and button). These properties range from color, dimensions, and positioning to a setting that enables or disables end-user interaction. Styling in this way enables you to determine the **appearance** of not only the controls, but also of the page as a whole. The screenshots below show the same two controls styled with two different sets of properties.





## How to use controls

All the controls that are available in MobileTogether are displayed in the Controls Pane[266]. To add a control to the design, drag it from the Controls Pane[266] and drop it at the required location in the page design. You can then do the following:

- *Link the control to a page source node* (available in the Page Sources Pane[270]) so that data from a node in the page source can be displayed in the design and processed. Typically, an association between a control and node is defined by dragging a data node from its source tree (in the Page Sources Pane[270]) onto the control. Such an association is called the **page source link** of the control.
- *Set control actions*: You can add one or more control actions[667] to execute when a control-related event[665] is triggered (for example, when a button is clicked). Define control actions[667] by right-clicking the control and selecting **Control Actions**. The available actions are described in the section Actions[667].
- *Set control properties:* The properties of a control define the control's appearance (including it position on the page). To set a control's properties, select the control and set its properties in the Styles & Properties Pane[274]. The properties of controls are described in the section Controls[404]. Note that it is the properties of the controls on a page that determine the appearance of the page.

## In this section

This section is organized as follows:

- Controls[404] describes each control separately: its function, its properties, and its event/s (for example, the button control[417] has one event, the `OnButtonClicked` event)
- Control Events[665] lists, in one table, the events of various controls

# 9.1 Controls

A page design (*screenshot below*) consists of page controls—such as combo boxes, tables, and images—that are laid out and formatted exactly as the end user will see the page. Controls are dragged into the design from the Controls Pane 266. After a control has been placed in the design, you can do the following:

- *Controls can be linked to page source nodes* (available in the Page Sources Pane 270) so that data from these nodes can be displayed in the design and processed. Typically, an association between a control and a page source node is defined by dragging a node from its source tree (in the Page Sources Pane 270) onto the control. A node association of this type is called the **page source link** of the control.
- *Set control actions*: Control actions 667 determine the functionality to be executed when a control-related event 665 is triggered (for example, when a button is clicked). You define control actions 667 by right-clicking a control, and selecting **Control Actions**. Control actions 667 are described in the section Actions 667.
- *Set control properties:* The properties of a control define the control's appearance (including position). Consequently, a control's properties also define the appearance of the page. To set a control's properties, select the control and set its properties in the Styles & Properties Pane 274. Each control's properties are described in the sub-section of this section.

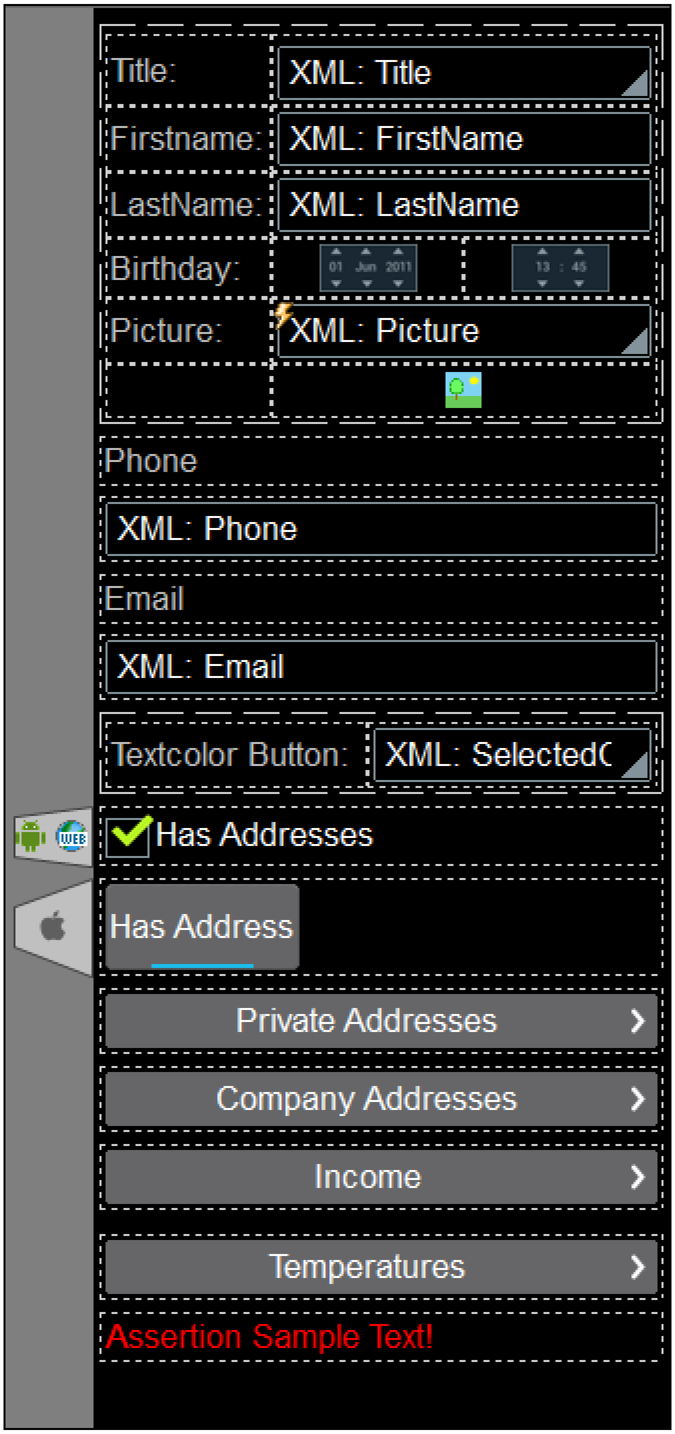
⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View 253) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog 296).
- A control's CSS properties can be defined in the Styles & Properties Pane 274 and/or in an external CSS file 296. Those defined in the Styles & Properties Pane 274 have priority.

## List of controls

Given below is a list of available controls, arranged alphabetically, together with a screenshot showing different controls..

- Assertion Message 409
- Button 417
- Chart 438
- Checkbox 447
- Combo Box 459
- Date 473
- DateTime (iOS) 484
- Edit Field 495
- Geolocation Map 508
- Horizontal Line 515
- Horizontal Slider 520
- HTML Label 529
- Image 541
- Label 554
- Placeholder 568
- Radio Button 571
- Rich Text 584
- Signature Field 590
- Space 600
- Switch 603
- Table 614
- Time 640
- Vertical Line 651
- Video 655



## Context menu

Each control in the page design has a context menu. The following control-related commands are common to most context menus.

▼ Unassign Page Source Link

⊟ *Description*

This command is enabled for those controls that can be associated with a page source node and for which an association exists. **Unassign Page Source <Name>** deletes the association between control and page source. Note that there is no other way to delete the control's association with a node.

▼ Control Actions

⊟ *Description*

Displays the Control Actions dialog[665], in which you can set actions for various control events. For a description of available actions for control events and how to set control actions, go to the section, Page Design | Actions[667].

▼ Control Variables

⊟ *Description*

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with the corresponding values till the control is called again. Parameters can be used in the XPath expression that calculates a variable's value.

Control variables are useful for providing values that can be used in control actions. Consider, for example, a `Department` node inside a `Company` node where the domain name of each `Company` node (for example, `altova.com` or `nanonull.com`) is provided in a child node of the `Company` node. If you now set a control variable to select the domain-name node, then the control variable can be used in an action of the control to dynamically select each company's domain name when that company is being processed. For example, the email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's `Control Variables` property (available in the Styles & Properties pane[274] when the control is selected). The currently defined control variables are listed as sub-items of the `Control Variables` property; each control variable can be edited by double-clicking it.

▼ Enter Text

⊟ *Description*

Enabled for controls in which text can be entered. Rolls out a sub-menu with the following options:

- *Directly:* To enter static text directly as the text of the control
- *XPath:* Displays the Edit XPath/XQuery Expression dialog[1244], in which you can enter the XPath expression that selects the text of the control
- *XML Node:* Refers to the option of displaying the content of an XML node as the control's text. Clicking the option displays a hint that a page source node can be dragged from the

Page Sources Pane[270] onto the control. The dragged-and-dropped node will be associated with the control, and the node's content will be entered as the text of the control

### Localization

#### *Description*

Displays the Localization dialog[1600], in which you can define the localization (translation) of strings that appear in various controls of the project. This command has the same effect as the Project | Localization[1600] command. For a description of localization, go to the description of the Project | Localization[1600] command.

### Deploy/Embed File

#### *Description*

If a control has a file that can be deployed or if an Image control[541] has a file that can be embedded, then the respective sub-commands of the **Deploy/Embed File** command are enabled.

- For the option to deploy a file, you can choose the sub-command for where you want to deploy the file. On deployment, the file is added to the *Deployable Files* list of the Files Pane[259].
- When you embed an image file, the binary data of the image file (PNG, BMP, etc) is converted into text-based Base64-encoding, and this text is embedded in the design file. On embedding, the Image control's[541] `Embed Image` property is set to `true`. See the description of the property for more information.

### Create Template from Selection

#### *Description*

If a control template[1200] can be created from the selected design component/s, then the **Create Template from Selection** command is enabled. Selecting the command does the following: (i) adds to the project a new control template that contains the selected design component/s (see the Pages Pane[257]); (ii) replaces the selected design component/s with a Placeholder Control[568]. To return to the previous state, use the context menu command **Replace Placeholder with Template Content** *(see description immediately below).*

### Replace Placeholder with Template Content

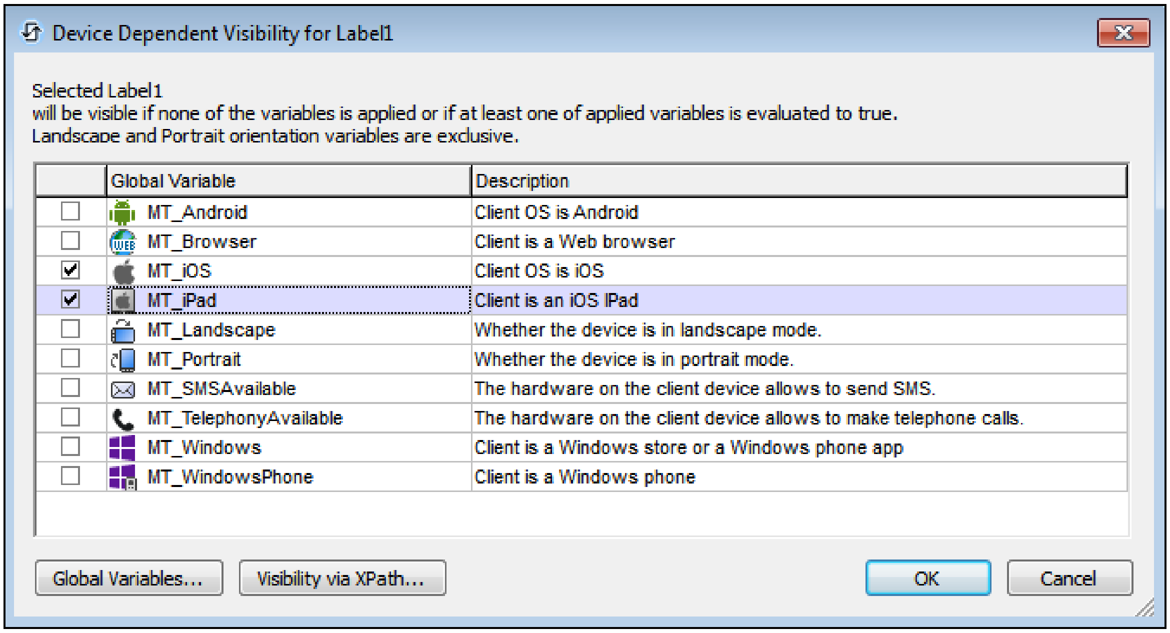
#### *Description*

This command is enabled when for Placeholder Controls[568], which are controls that instantiate a control template[1200] of the project. The command replaces the Placeholder Control[568] with the content of the control template[1200] for which it is a placeholder. Note that neither the control template nor its content is deleted. The reverse command is **Create Template from Selection** *(see description immediately above).*

### Device Dependent Visibility

#### *Description*

Displays the Device Dependent Visibility dialog (*screenshot below*). The dialog contains a list of client-device types. Select the client-device type for which you want the control to be visible, and click **OK**.

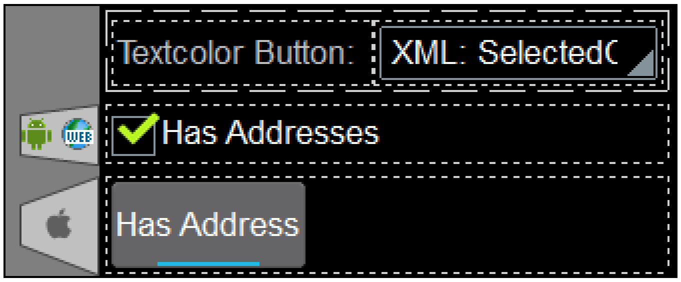


Device Dependent Visibility for Label1

Selected Label1
will be visible if none of the variables is applied or if at least one of applied variables is evaluated to true.
Landscape and Portrait orientation variables are exclusive.

| | Global Variable | Description |
|---|---|---|
| □ | MT_Android | Client OS is Android |
| □ | MT_Browser | Client is a Web browser |
| ☑ | MT_iOS | Client OS is iOS |
| ☑ | MT_iPad | Client is an iOS IPad |
| □ | MT_Landscape | Whether the device is in landscape mode. |
| □ | MT_Portrait | Whether the device is in portrait mode. |
| □ | MT_SMSAvailable | The hardware on the client device allows to send SMS. |
| □ | MT_TelephonyAvailable | The hardware on the client device allows to make telephone calls. |
| □ | MT_Windows | Client is a Windows store or a Windows phone app |
| □ | MT_WindowsPhone | Client is a Windows phone |

If the client-device type has an icon, this icon appears in the design, to the left of the control to which it applies (*see screenshot below*).



### Page Actions

#### *Description*

Displays the Page Actions dialog[1631], in which you can set actions for various page events. This command has the same effect as the Page | Page Actions[1631] command. For a description of available actions for page events and how to set page actions, go to the description of the Page | Page Actions[1631] command.

### Actions Overview

⊟ *Description*

Displays the Actions Overview dialog 1632 of the currently active page. This command has the same effect as the Page | Actions Overview 1632 command. For more information, go to the description of the Page | Actions Overview 1632 command.

## 9.1.1 Assertion Message

The Assertion Message control displays the assertion message of the first invalid assertion of the page. An `Assertion` is a property of a page and of some—not all—controls. It specifies a certain condition (for example that a node may not be empty). If the `Assertion` property's condition is not met, the assertion is invalid, and the **`Assertion Message` property** associated with that `Assertion` property is displayed in the **Assertion Message control**.

An Assertion Message control can be placed anywhere in the design. It will always display the `Assertion Message` property text that is associated with the first invalid assertion on the page. If there are multiple invalid assertions on a page, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. The Assertion Message control should therefore be inserted only once on a page. If multiple Assertion Message controls are placed in the design, they will all display the same assertion message (that of the first invalid assertion).

Assertions and assertion messages work as follows:

- The `Assertion` property of a control or page sets a condition to be met in order for the assertion to be valid. The assertion's condition is specified with an XPath expression.
- If the assertion is invalid, then the text of the control's `Assertion Message` property is displayed in the Assertion Message 409 control.

For example: The XPath expression `LastName != ""` in the `Assertion` property of a control asserts that the node `LastName` must not be empty. If this node is empty, then the control's assertion message is displayed in the page at the point where the Assertion Message 409 control is inserted.

⊟ Notes

- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

### Assertion Message events

There is no event associated with the Assertion Message control.

## Assertion Message properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"          : $XML1/R/@bold = "1",
    "Italic Text"        : true(),
    "Text"               : "hello",
    "Text Color"         : "red",
    "Background Color"   : $XML1/R/@background,
    "Text Size"          : $XML1/R/@textsize
   }

map{
    "Style Sheet"        : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Multiline

Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.

- If this property is set to **true** on a label control, the Max Number of Lines property becomes available.
- Check boxes with multilines can be vertically aligned via the Vertical Alignment property.

▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a text color for when the control is disabled, use the **Text Color (Disabled)** property.

▼ Text Size

Select a size from the dropdown list of the combo box. Allowed values are: smallest|small|medium|large|largest. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value medium.

You can generate other values by using the mt-font-height[1262] function. For example, to get a size that is 120% larger than the numeric size that corresponds to **'largest'** on a device, use the following XPath expression for the **TextSize** value: mt-font-height('largest', 'px') * 1.2. The function generates the numeric (pixel) value that corresponds to the **'largest'** size. This value is then multiplied by **1.2** to obtain the numeric value that is 120% of the value that corresponds to **'largest'**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is  the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template1200 to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value

is selected, the property `Max Control Width` becomes available

- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-

independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.2 Button

Buttons can be used to execute an action when the button is clicked. The name of the button can be static text (entered as the value of the `Text` property; *see below*) or a dynamic value obtained from a page source node (by dragging the node onto the button). You can also alternatively or additionally select an icon from the options available in the `Button Image` property combo box. The height of the button is determined by (i) the height of the button icon or text, whichever is greater, and (ii) the vertical padding set on the button. The `OnButtonClicked` event is associated with the button control. To define an action for this event, click the **Additional Dialog** button of the `Control Action` property. This displays the Control Actions dialog 667, in which you can specify the required action.

⊟ Notes

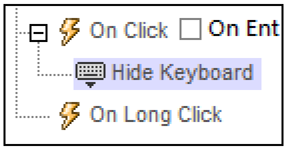
- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View 253) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog 296).
- A control's CSS properties can be defined in the Styles & Properties Pane 274 and/or in an external CSS file 296. Those defined in the Styles & Properties Pane 274 have priority.

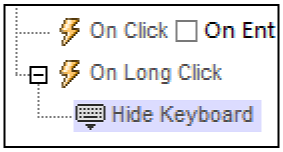
### Button events

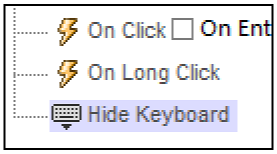
The **`OnButtonClicked`** event is available. To define actions for the button's **`OnButtonClicked`** event, right-click the button and, from the context menu that appears, select **Control Actions for OnButtonClicked**. This displays the Actions dialog for button events. For a description of the actions that can be defined for this event, see the Actions section 667.

▼ OnButtonClicked (OnClick, OnLongClick)

The end user can click the control in one of two ways: a short tap (`On Click`) or a longer press (`On Long Click`). A sequence of different actions 667 can be specified for each type of click (*see screenshots at left and middle below*). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions 667 be executed after those of the end-user click; these actions are defined after the `On Long Click` event (*see screenshot below right*)..

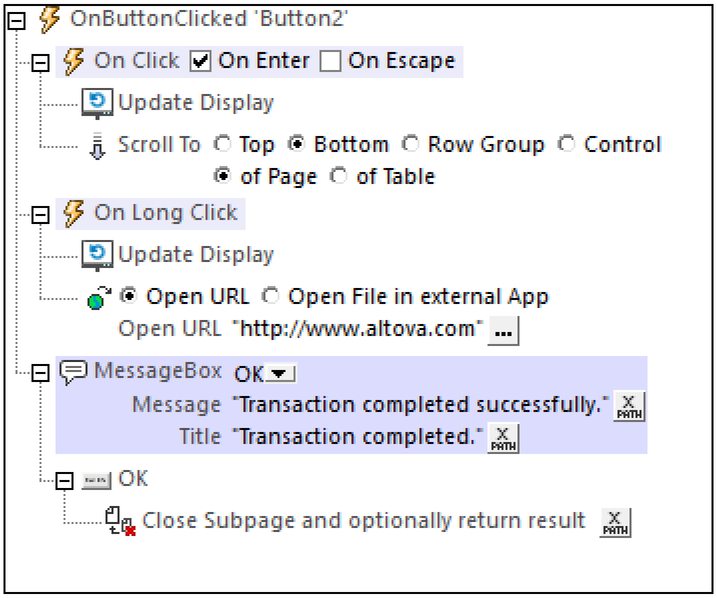

⊟ On Click □ On Ent
Hide Keyboard
On Long Click


On Click □ On Ent
⊟ On Long Click
Hide Keyboard


On Click □ On Ent
On Long Click
Hide Keyboard

- `On Click`: The action/s to perform when the control is tapped (*see screenshot above left*).
- `On Long Click`: The action/s to perform when the control is pressed for a longer time than a tap (*see screenshot above center*).
- *Additional actions:* The action/s to perform after the `On Click` or `On Long Click` actions have been executed (*see screenshot above right*). If no action has been set for the `On Click` or `On Long Click` events, then the additional action/s are performed directly on a click or long-click.

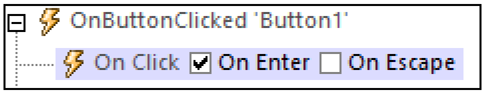
You can combine actions [667] for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.


⊟ OnButtonClicked 'Button2'
⊟ On Click ☑ On Enter □ On Escape
Update Display
Scroll To ○ Top ● Bottom ○ Row Group ○ Control
● of Page ○ of Table
⊟ On Long Click
Update Display
● Open URL ○ Open File in external App
Open URL "http://www.altova.com" ...
⊟ MessageBox OK
Message "Transaction completed successfully."
Title "Transaction completed."
⊟ OK
Close Subpage and optionally return result

The screenshot above shows that the `On Click` and `On Long Click` events each has a sequence of actions defined for it. An additional `MessageBox` event is defined after the `On Long Click` event. This `MessageBox` event will be executed after the `On Click` or `On Long Click` sequence of actions has completed.

### *On Enter/Escape*

If the control's *On Enter* or *On Escape* check box is selected, then the control's actions are executed when the respective key (**Enter** or **Escape**) is tapped. The key-tap (**Enter** or **Escape**) serves as an alternative to the `On Click` event, and will work additionally to the click. The screenshot below shows the *On Enter* and *On Escape* check boxes of the Button event. Other controls that provide this option look similar and work similarly.


⊟ OnButtonClicked 'Button1'
On Click ☑ On Enter □ On Escape

This setting can also be accessed via the control's `On Enter/Escape` property, which is described below.

**Note:** If you select the **Page | Show/Define Tab Order**[1633] menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

## Button properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Text

The `Text` property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the Rich Text[584] control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text[1227] section for more information.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

▼ Multiline

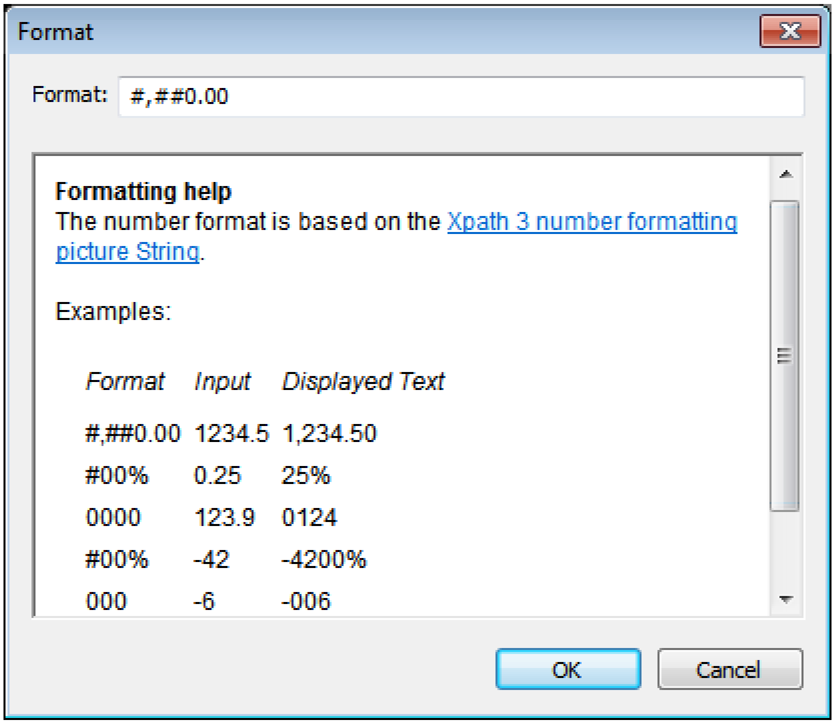
Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

▼ Number Format String

Click the **Additional Dialog** button and enter a number format in the Format dialog that appears (*screenshot below*).



Format: #,##0.00

**Formatting help**
The number format is based on the Xpath 3 number formatting picture String.

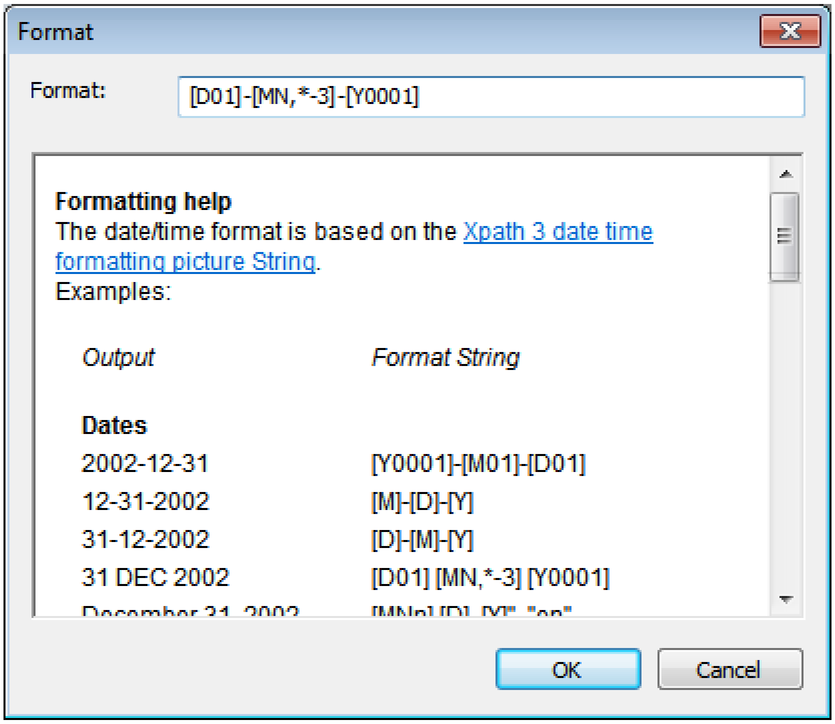
Examples:

| *Format* | *Input* | *Displayed Text* |
|---|---|---|
| #,##0.00 | 1234.5 | 1,234.50 |
| #00% | 0.25 | 25% |
| 0000 | 123.9 | 0124 |
| #00% | -42 | -4200% |
| 000 | -6 | -006 |

The formatting will be applied to the control's content if the content is numeric, and will be displayed in the solution, not in the design.

▼ Date/Time Format String

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (*screenshot below*).



Format: [D01]-[MN,*-3]-[Y0001]

**Formatting help**
The date/time format is based on the Xpath 3 date time formatting picture String.
Examples:

| *Output* | *Format String* |
|---|---|
| **Dates** | |
| 2002-12-31 | [Y0001]-[M01]-[D01] |
| 12-31-2002 | [M]-[D]-[Y] |
| 31-12-2002 | [D]-[M]-[Y] |
| 31 DEC 2002 | [D01] [MN,*-3] [Y0001] |
| December 31, 2002 | [MNn] [D], [Y]", "en" |

The formatting will be applied to the control's content if the content has the correct lexical form of `xs:date` (for the Date control), `xs:time` (for the Time control), or `xs:dateTime` (for the Date, Time, and DateTime controls). Basic examples are:

- *xs:date:* `2014-12-31`
- *xs:time:* `23:59:59`
- *xs:dateTime:* `2014-12-31T23:59:59`

### Date/Time Format Language

Select one of the supported languages from the dropdown list of the combo box (`EN`, `DE`, `ES`, `FR`, `JA`). The selected language will be used in the date/time formatting that is set in the `Date/Time Format String` property (*see description above*). If the names of months and weekdays are used in the format string, then these will be displayed in the language selected for this property. The default language is English.

### Button Image

Adds a predefined icon or a custom image to the button display. You can also optionally specify (in the `Text` property) an additional text string to accompany the button icon. If you use both icon and text, then the `Button Image Position` property specifies whether the icon is positioned to the left or the right of the text, and the `Button Image/Text Distance` property specifies the distance in pixels between the button and the text. Horizontal positioning of the icon-text pair within the button is specified with the `Horizontal Alignment` property.

Add a predefined icon by selecting one from the dropdown list of the property's combo box. Available icons include: +, –, >, *Barcode, Calendar, Cancel, Close, Closed Tree, Copy, Cut, Delete, Dragging, Dragging Popup, Edit*, *Email, Export, Fast Forward, Fast Rewind, Help, Import, Link*, *Microphone, Offline, OK, Opened Tree, Paste, Pause, Photo, Photo Gallery, Play, Play Reverse, Print, Print PDF, Print Word,* *Redo, Refresh, Report, Resume, Search, Select Multiple, Select Single, Settings, Share, Snooze, Stop, Stop Playing, Thumbs Up, Thumbs Down, Time, Undo, Unlink, View* and *Web*.

Add a custom image by selecting *Custom Image*. On doing this, two related properties become available: `Image Source` and `Image Source Type` (*see their descriptions below*).

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is any one of the values listed above. *(For users of localized (non-English) MobileTogether Designer editions: Note that the expression must evaluate to the English-language value, not the localized value.)*

The default value of the `Button Image` property is no value, which results in no icon being displayed.

When a predefined icon or *Custom Image* is selected, then the `Button Image Color`, `Button Image Color (Disabled)`, `Button Background`, `Button Image Position`, and `Button Image/Text Distance` properties become available. These enable you to set, respectively, (i) a transparent or non-transparent background for the button, and (ii) the horizontal position of the icon relative to button text.

▼ Button Image Color

The `Button Image Color` property is enabled when the `Button Image` property has been set to a predefined image or custom image. It sets the color of the button image when the button is enabled. The default value is client-specific.

▼ Button Image Color (Disabled)

The `Button Image Color (Disabled)` property is enabled when the `Button Image` property has been set to a predefined button image or custom image. It sets the color of the button image when the button is disabled. The condition/s for enabling and disabling a button is set on the button's `Enabled/Editable` property. The property's default value is client-specific.

▼ Button Background

Selects whether the background of the button icon is transparent or non-transparent. The default is `not transparent`.

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is either `transparent` or `not transparent`. Note that you cannot set padding on transparent buttons. *(For users of localized (non-English) MobileTogether Designer editions: Note that the expression must evaluate to the English-language value, not the localized value.)*

▼ Button Image Position

Specifies whether the button image (predefined icon or custom image) is located to the left or right of the button text (which is specified via the `Text` property). The default value is `left of text`.

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is either `left of text` or `right of text`. *(For users of localized (non-English) MobileTogether Designer editions: Note that the expression must evaluate to the English-language value, not the localized value.)*

▼ Button Image/Text Distance

Sets the distance, in pixels, dp, or sp between the button image and button text. The default value is the default button padding (which is `0px`).

The value of this property can also be entered as an XPath expression that is a string value. The allowed string values are those listed in the property's combo box.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Image Source

The value of the `Image Source` property references an image in one of the following ways:

- The URL of a binary image file (`PNG`, `BMP`, etc). The value of the property must be a URL. The URL is selected in the Specify File dialog (*see description below*).
- An image file represented as a Base64-encoded string. The value of the property must be a Base64-encoded string. An XPath expression supplies the string, which could be entered directly or obtained form an XML node.
- An SQL SELECT statement that queries a page source on the server. The query should return the Base64-encoded string that is to be used as the image. The SELECT statement is generated by an XPath expression.

The type of image source is provided by the property `Image Source Type` (*see next property below*). By default, `Image Source Type` is set to `url`. The `Image Source` property automatically opens the corresponding dialog: Specify File dialog for `url` *(see below)*, and Edit XPath XQuery Expression dialog 1244 for `base64` (*see Base64-Encoded Images* 1098).

**Note:** If the image source is a URL and the URL is changed during simulation or while the solution runs, then the image must be explicitly reloaded with the Reload action 801. For example, if a combo box selection changes the selection of an image, a Reload action 801 targeting the image must be defined on the combo box.

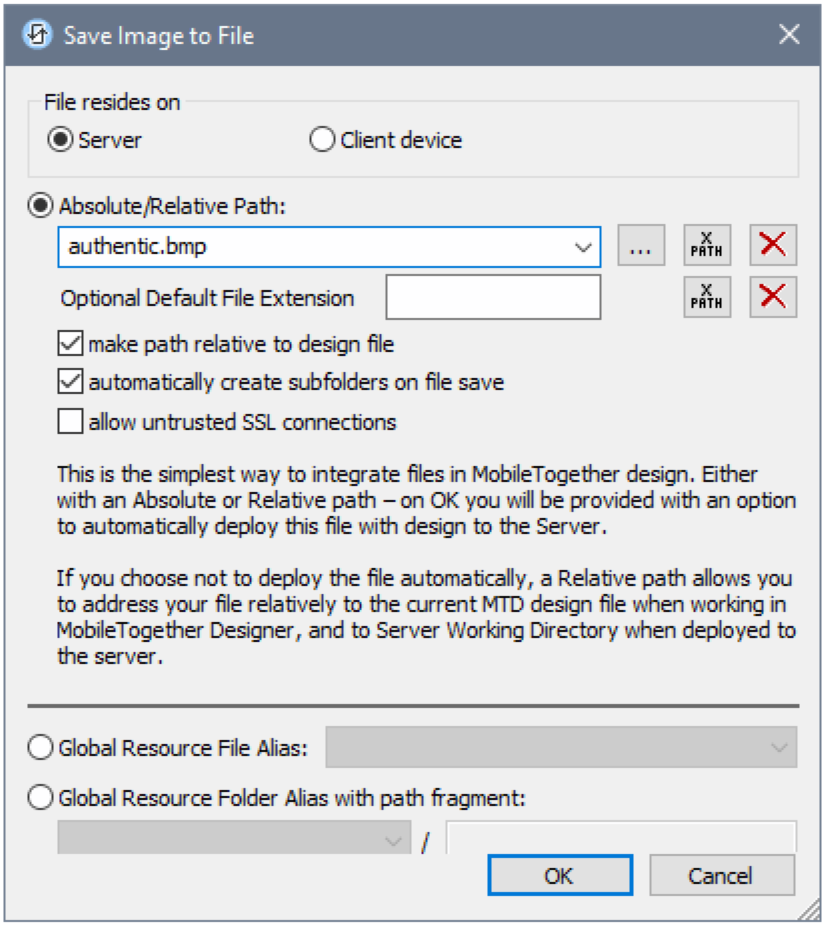
⊟ *The Specify File dialog*

You can choose a file on the server or the client. Select the respective radio button option and, as required, enter details and select options (*see below*).

After finishing and clicking **OK**, a dialog appears asking whether you want to deploy the file to the server and/or client and whether you want to embed the image in the design file.

- *Deploy to Server:* The fie will be downloaded from server to client when the solution is run. This could add to the time-overhead when running the solution.
- *Deploy to Client:* The file is deployed to the client at the time the solution is downloaded.
- *Embed Image in Design File:* The image is embedded in the design file (in Base64 format). This option is useful when you want to share the design file without having to share the image file and the folder structure that contains the file. Selecting this automatically set the **`Embed Image`** 541 property to `true`. Note, however, that for deployment, you still need to specify where the file is to be deployed (server and/or client).

*File is located on server*
If the image file is located on the server, you can either browse for it (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want (*see screenshot below*).



Save Image to File

File resides on

● Server ○ Client device

● Absolute/Relative Path: authentic.bmp

Optional Default File Extension

☑ make path relative to design file

☑ automatically create subfolders on file save

□ allow untrusted SSL connections

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:

○ Global Resource Folder Alias with path fragment:

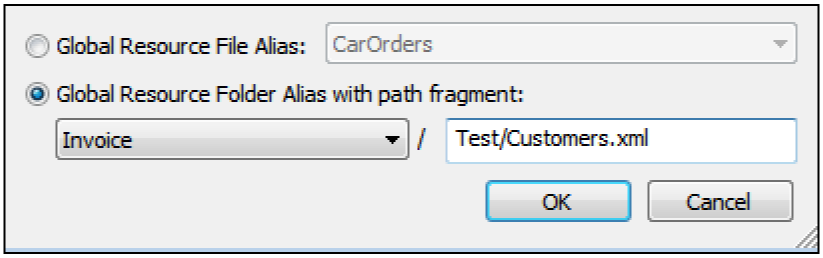
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). See the section Altova Global Resources[1334] for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

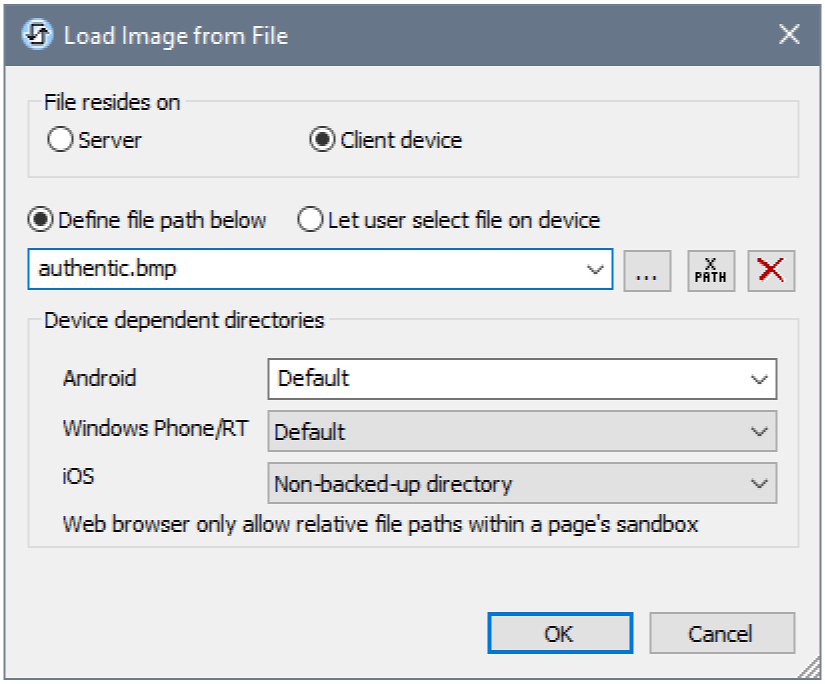
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources[1334] for details.

*File is located on client*

If the image file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



Load Image from File

File resides on

○ Server ● Client device

● Define file path below ○ Let user select file on device

authentic.bmp

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To[686] (*Source File* and *Target File* options), Load/Save File[809], Load/Save Image[707], Load/Save Binary File[815], Load/Save Text File[821], Read Folder[847], and Get File Info[849].

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.
- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for

files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog 1663 (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

_Filename is defined by the end user (on the client device)_

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html','xml'`).
- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.
- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

▼ Image Source Type

Sets the type of the image source selected by the `Image Source` property above. Three type options are available:

- `url`: a binary image file, such as a `PNG` or `BMP` image file; this is the default image source type
- `base64`: a Base64-encoded string to be used as the image
- `SQL`: an SQL SELECT statement that queries the page source on the server and returns a Base64-encoded string to be used as the image

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is a URL, a Base64 string, or an SQL SELECT statement.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog[667]. You can set actions to perform when a control event[665] is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog[667]. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Instant Row Dragging

This property applies to buttons located in a row group of a top-level table. Set this property to `true` if you want to allow the row containing the button to be moved immediately the button is dragged. The row can be moved to a new location within the row group, effectively changing the position of the row within the row group. The property's default value is `false`. If this button property is set to `false` and the row group has been defined as being draggable, then the button must be long-tapped to activate Drag mode. If the property's value is set to `true`, then dragging starts immediately. Note that in this case the tap and long-tap are disabled and also the **`OnButtonClicked`** actions. See the topic Table Properties[1078] for more information.

The property's value can also be generated dynamically via an XPath expression.

### ▾ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

### ▾ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

### ▾ Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` 1262 function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically

maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**1304 and **`$MT_CanvasY`**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is  the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template1200 to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Strikethrough Text

Select `true` or `false` from the dropdown list of the combo box to determine whether a line is drawn through the text of the control. The value can also be generated dynamically via an XPath expression. The default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

### Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX,`

$MT_CanvasY).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as high as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as high as the control's content requires.

In effect, `fill_parent` creates a maximum height, while `wrap_content` creates a minimum height.

*Default values*

- For the the Geolocation Map control, the default is the smaller of the two values **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX,`

$MT_CanvasY).

- For all other controls, the default is wrap_content.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**1304 and **$MT_CanvasY**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order**1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ On Enter/Escape

Takes one of three values:

- `OnEnter`: Specifies that the actions of this control are executed when the **Enter** key is tapped.
- `OnEscape`: Specifies that the actions of this control are executed when the **Escape** key is tapped.
- None: No action when either **Enter** or **Escape** is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to **`"OnEnter"`** or **`"OnEscape"`**. If more than one control on a page is given the same value (`OnEnter` or `OnEscape`), then the first visible and enabled control that has the value is selected when the key is tapped. (See the **`Visible`** and **`Enabled/Editable`** properties.)

This setting can also be made via the dialog to set the control's **`OnClicked`** actions (see the description of the control's events above).

**Note:** If you select the **Page | Show/Define Tab Order**1633 menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.3 Chart

The Chart control enables data from a source data file to be displayed in the form of a chart. The available chart types 1171 are: pie charts, bar charts, line graphs, area charts, candlestick charts, and gauge charts. Data for the X-Axis, Y-Axis, and other chart components is selected with XPath expressions. The context node for these XPath expressions is set by dragging it from the source data tree and dropping it onto the chart control in the design.

The chart's display settings within the page are defined in the Styles & Properties Pane 274. The settings for chart type, data selection, and appearance are defined in the Chart Configuration dialog. This dialog is accessed by clicking the **Additional Dialog** button of the `Chart Settings` property, or by double-clicking the chart in the design.

For detailed information about how to configure charts, see the Charts 1171 section.

⊟ Notes

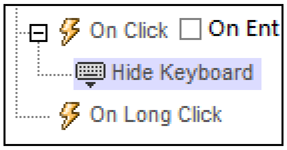
- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View 253) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog 296).
- A control's CSS properties can be defined in the Styles & Properties Pane 274 and/or in an external CSS file 296. Those defined in the Styles & Properties Pane 274 have priority.

## Chart events

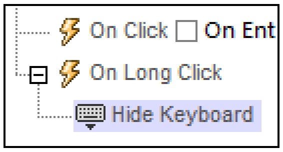
The **`OnimageClicked`** event is available (the image that is clicked is the chart). To define actions for the chart's **`OnImageClicked`** event, right-click the chart and, from the context menu that appears, select **Control Actions for OnImageClicked**. This displays the Actions dialog for chart events. For a description of the actions that can be defined for this event, see the Actions section 667.

▼ OnChartClicked (OnClick, OnLongClick)

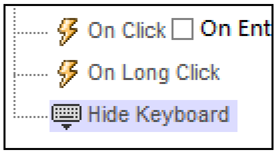
The end user can click the control in one of two ways: a short tap (`On Click`) or a longer press (`On Long Click`). A sequence of different actions 667 can be specified for each type of click (*see screenshots at left and middle below*). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions 667 be executed after those of the end-user click; these actions are defined after the `On Long Click` event (*see screenshot below right*)..



On Click □ On Ent
Hide Keyboard
On Long Click



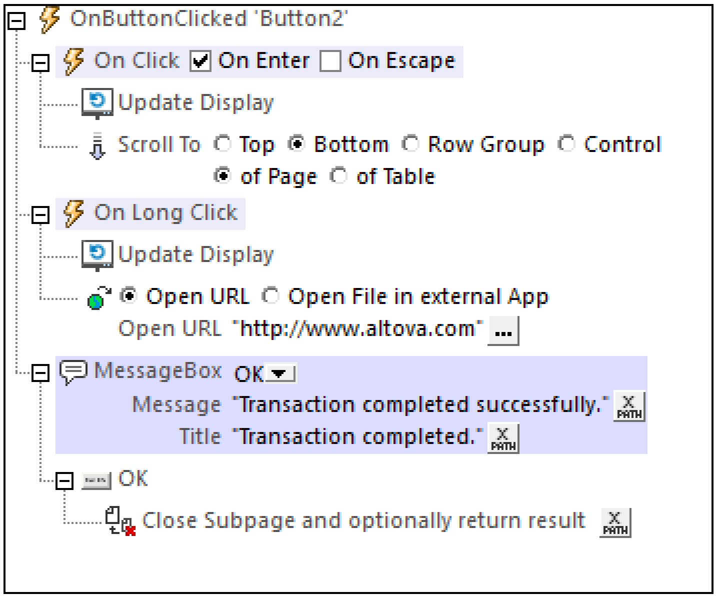
On Click □ On Ent
On Long Click
Hide Keyboard



On Click □ On Ent
On Long Click
Hide Keyboard

- `On Click`: The action/s to perform when the control is tapped (*see screenshot above left*).
- `On Long Click`: The action/s to perform when the control is pressed for a longer time than a tap (*see screenshot above center*).
- *Additional actions:* The action/s to perform after the `On Click` or `On Long Click` actions have been executed (*see screenshot above right*). If no action has been set for the `On Click` or `On Long Click` events, then the additional action/s are performed directly on a click or long-click.

You can combine actions 667 for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.

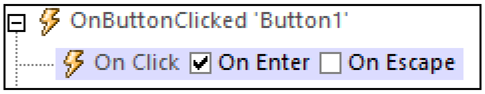


OnButtonClicked 'Button2'
On Click ☑ On Enter □ On Escape
Update Display
Scroll To ○ Top ● Bottom ○ Row Group ○ Control
● of Page ○ of Table
On Long Click
Update Display
● Open URL ○ Open File in external App
Open URL "http://www.altova.com"
MessageBox OK
Message "Transaction completed successfully."
Title "Transaction completed."
OK
Close Subpage and optionally return result

The screenshot above shows that the `On Click` and `On Long Click` events each has a sequence of actions defined for it. An additional `MessageBox` event is defined after the `On Long Click` event. This `MessageBox` event will be executed after the `On Click` or `On Long Click` sequence of actions has completed.

*On Enter/Escape*

If the control's *On Enter* or *On Escape* check box is selected, then the control's actions are executed when the respective key (**Enter** or **Escape**) is tapped. The key-tap (**Enter** or **Escape**) serves as an alternative to the `On Click` event, and will work additionally to the click. The screenshot below shows the *On Enter* and *On Escape* check boxes of the Button event. Other controls that provide this option look similar and work similarly.



OnButtonClicked 'Button1'
On Click ☑ On Enter □ On Escape

This setting can also be accessed via the control's `On Enter/Escape` property, which is described below.

**Note:** If you select the **Page | Show/Define Tab Order**[1633] menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

## Chart properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Chart Settings

Click the **Additional Dialog** button to display the Chart Configuration dialog. The settings you make in this dialog will apply to the chart that is currently selected in the design. For a description of how to configure charts, see the section, Charts[1171].

▼ ID

This property must be entered when the chart is placed in a repeating table[1065] or repeating row of a dynamic table[1069]. The value of the ID property can be any string, but must evaluate to a different ID for each instantiated chart. This can be achieved by assigning a dynamic XPath expression as the value of the property.

▼ Create Before Load

In the combo box, select the value you want: `true` or `false`. If `true`, the chart or Base64 image is created before the page loads. If `false`, a page sources action must be used to create the chart or image. The default value is `true`.

▼ Chart Creation Width

Sets the width, in pixels, of the chart to be generated. Click the **Edit XPath** icon and, in the dialog that appears, enter an expression that returns a numeric value. This value will be the width, in pixels, dp or sp, of the chart to be generated.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Chart Creation Height

Sets the height, in pixels, of the chart to be generated. Click the **Edit XPath** icon and, in the dialog that appears, enter an expression that returns a numeric value. This value will be the height, in pixels, dp or sp, of the chart to be generated.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-

independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog[667]. You can set actions to perform when a control event[665] is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog[667]. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Limit Control Height to Canvas

Select one of the allowed values (`true` or `false`) in the combo box. In case the control's height exceeds the device height, a value of `true` restricts the height to that of the device display. Default is `true`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

On Enter/Escape

Takes one of three values:

- `OnEnter`: Specifies that the actions of this control are executed when the **Enter** key is tapped.
- `OnEscape`: Specifies that the actions of this control are executed when the **Escape** key is tapped.
- None: No action when either **Enter** or **Escape** is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to **`"OnEnter"`** or **`"OnEscape"`**. If more than one control on a page is given the same value (`OnEnter` or `OnEscape`), then the first visible and enabled control that has the value is selected when the key is tapped. (See the **`Visible`** and **`Enabled/Editable`** properties.)

This setting can also be made via the dialog to set the control's **`OnClicked`** actions (see the description of the control's events above).

**Note:** If you select the **Page | Show/Define Tab Order** 1633 menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu[404].

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.4 Check Box

Check boxes allow one of two possible values to be entered as the content of a node. In this way the user can be constrained to select one of two specific values. When you insert a check box control, you can specify whether the check box should be to the left or right of the check box text, or in the default system position. Two key properties of the check box control are:

- The text that accompanies the check box. This can be static text (entered as the value of the `Text` property; *see below*) or a dynamic value obtained via an XPath expression.
- The values that are to be respectively assigned the checked and unchecked states of the check box. These are assigned with the `Checked Values` property (*see below*). To specify which node will receive the value, make a page source link from the check box to a page source node (by dragging the node on to the check box).

Check boxes have the `OnFinishEditing` event, which is triggered when the end-user makes a check box selection. To define an action for this event, click the **Additional Dialog** button of the `Control Action` property. This displays the Control Actions dialog[667], in which you can specify the required action.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View 253) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog 296).
- A control's CSS properties can be defined in the Styles & Properties Pane 274 and/or in an external CSS file 296. Those defined in the Styles & Properties Pane 274 have priority.

## Check Box events

The **`OnFinishEditing event`** 665 is available. For a description of the actions that can be defined for this event, see the Actions section 667.

## Check box properties

The control's properties are available in the Styles & Properties Pane 274, and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog 296 has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"          : $XML1/R/@bold = "1",
    "Italic Text"        : true(),
    "Text"               : "hello",
    "Text Color"         : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Text

The `Text` property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the Rich Text[584] control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text[1227] section for more information.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

▼ Multiline

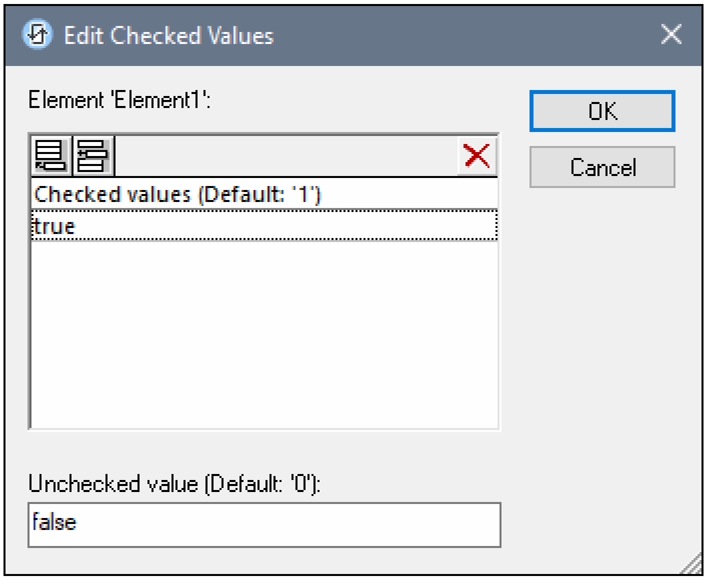
Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

## Checked/Unchecked Values

Provides an XML data value for the selected/unselected state of the control. The defaults are, respectively, **`1`** and **`0`**. To change the XML values that will be entered in the associated page source node, click the property's **Additional Dialog** button. In the Edit Checked Values dialog that appears (*screenshot below*), enter a value for each the checked (selected) and unchecked (unselected) state. When the end user changes the control's state, the value corresponding to the new state (selected/unselected) will be entered in the page source node.



Element 'Element1':

Checked values (Default: '1')

true

Unchecked value (Default: '0'):

false

Note the following points:

- To specify which node will receive the value, make a page source link from the control to a page source node (by dragging the node on to the control).
- If you enter more than one value for the *Checked* value, then the first value will be used; the others are ignored.
- If you enter a *Checked* value but do not enter an *Unchecked* value, then the *Unchecked* value is no longer the default value but an empty string.
- You can enter values for the *Checked/Unchecked Values (true/false)* property in a style sheet 1318 for: (i) all check boxes, and/or (ii) all switches, and/or (iii) all controls that use *Checked/Unchecked Values (true/false)* (which are check boxes and switches).
- If values for *Checked/Unchecked Values (true/false)* are defined in a style sheet 1318, and a control is, as a consequence, assigned values for this property in more than one place, then the most local definition wins. The most local definition is the style defined directly in the control's property. For priority involving a style sheet, see Priority within a Style Sheet 1321 and Priority across Style Sheets 1325.

## Get Value from XPath

The value returned by the XPath expression is displayed in the control. This enables alternative ways to enter display values for certain controls. For example, a combo box could take its display value from either a page source node or the return value of the **`Get Value from XPath`** property.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the generation of the value of the `Get Value from XPath` property. If used, then a validation error results.

### Auto Correct Value

The control has two states: checked and unchecked, each of which is associated with an XML value. These XML values are defined in the `Checked Values` property. The control is also associated with a page source node. If the value in the page source node is anything other than the XML value defined for the current status of the control (checked or unchecked), then auto-correction updates the value in the node to the value that has been defined for the current status.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node[886] action. The `Auto Correct Value` property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

For example, if the checked value is defined to be **`1`** and the unchecked value to be **`0`**, and if the node's content is **`"somestring"`**, then, (i) if the control is checked, the node's content will be corrected to **`1`**, (ii) if the control is unchecked, the node's content will be corrected to **`0`**. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction.

The `Auto Correct Value` property has two possible values: `true` or `false`. If the property is set to `true`, XML values are automatically corrected. The property's default value is `false`.

### Check Mark Position

Sets the position of the check box relative to the control's text: either to the left or right of the text. The default is the operating system default.

### Control Action

Click the **Additional Dialog** button to display the control's Actions dialog[667]. You can set actions to perform when a control event[665] is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog[667]. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

▼ Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height`[1262] function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is  the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template[1200] to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Strikethrough Text

Select `true` or `false` from the dropdown list of the combo box to determine whether a line is drawn through the text of the control. The value can also be generated dynamically via an XPath expression. The default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to

select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value

directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### ▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

### ▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically

maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** 1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.5 Combo Box

The Combo Box control provides a mechanism consisting of two parts:

- *Entry text and XML values*: The text of combo box entries (the dropdown list that the end user sees), and their corresponding XML values (that go into the page source node) are specified with the `Combo Box Entry Values` property (*see below for details*).
- *Associated XML node*. A node from the Page Sources Pane[270] is dropped on the combo box. This is the associated XML node (or page source link) which will receive the combo box value that is selected by the end user. In order to use the end-user-selected value elsewhere in the page, the node's content is referenced by either dragging the node onto controls or selecting the node in XPath expressions.

Typically, combo boxes allow users to select one option from out of a set. If you wish to offer users the ability to select multiple options, set the **`Multi Select`** property to **`true`**. For details of how this works, see the description of the property below.

If you want to have the dropdown list of a combo box open automatically when the user tabs to the combo box, then set the `Browser CSS Class` property of the combo box (*see properties below*) to a value of `mt-combo-open-on-focus`. If you want to set this behavior in a single place for multiple combo boxes, then set `mt-combo-open-on-focus` on the `Browser CSS Class` property of the Table[614] or Page[384] that contains the combo boxes.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.

- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

## Combo Box events

The **`OnFinishEditing event`**[665] is available. For a description of the actions that can be defined for this event, see the Actions section[667].

## Combo box properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Multiline

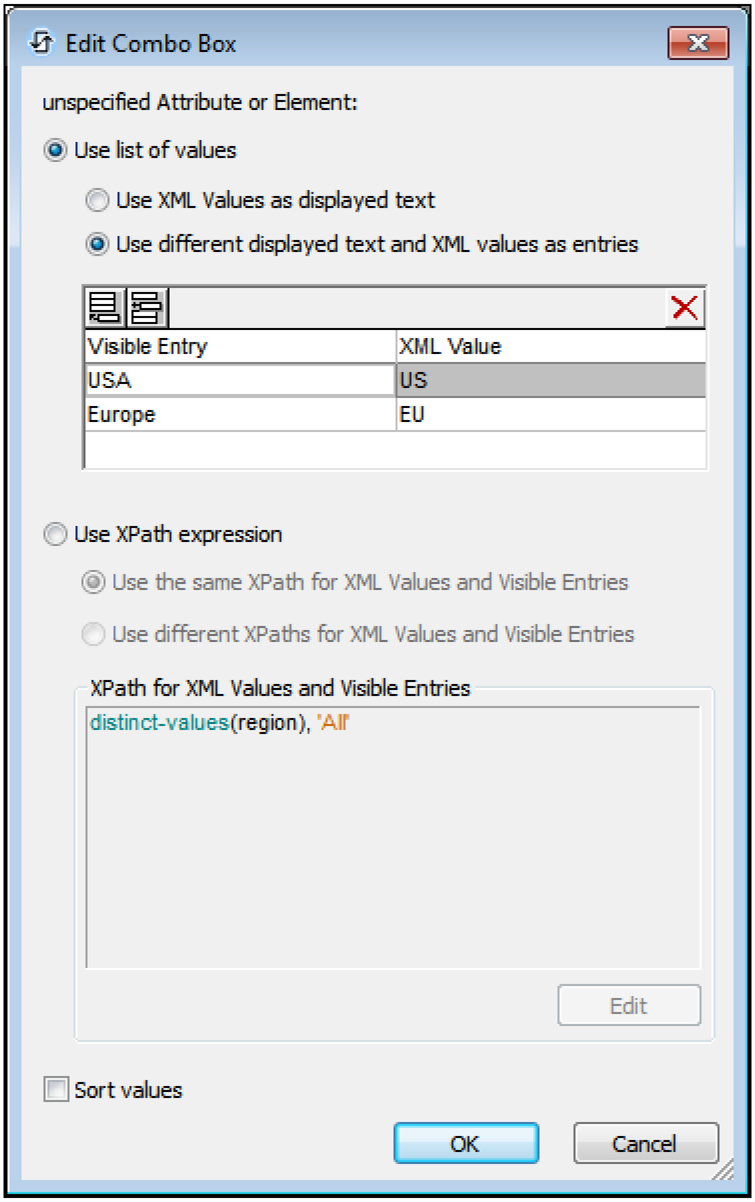
Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

▼ Combo Box Entry Values

Provides XML data values for the items of the dropdown list of the combo box. Click the **Additional Dialog** button to display the Edit Combo Box dialog (*screenshot below*). Alternatively, you can double-click the combo box to display the Edit Combo Box dialog.


Edit Combo Box
unspecified Attribute or Element:
● Use list of values
○ Use XML Values as displayed text
● Use different displayed text and XML values as entries

| Visible Entry | XML Value |
|---|---|
| USA | US |
| Europe | EU |

○ Use XPath expression
● Use the same XPath for XML Values and Visible Entries
○ Use different XPaths for XML Values and Visible Entries
XPath for XML Values and Visible Entries

```
distinct-values(region), 'All'
```

Edit
□ Sort values
OK
Cancel

To define the entries and values for the combo box, do the following:

1. Select the method with which you wish to define the entries and values by clicking the appropriate radio button to select values: (i) *Use list of values*, or (ii) *Use XPath expression*. How to use these methods is described below.
2. If you wish to have the items that appear in the drop-down list of the combo box sorted, check the *Sort Values* check box.
3. Click **OK** to finish.

*Using a list of values*
If you select *Use List of Values*, you can specify each item of the dropdown list of the combo box as a

text string. You can choose to use the text string for both (i) the text of the item in the dropdown list, as well as (ii) the XML data value of the item. Alternatively, you can specify different text strings for the dropdown list item and for its corresponding XML data value. For each item, add a new row and enter the desired text string/s.

*Using XPath expressions*
If you select *Use XPath expression*, you can use XPath expressions to specify the dropdown list items and the items' corresponding XML data values. There are two alternative ways of doing this:

- Use one XPath expression to define both the visible text (of the item in the dropdown list of the combo box) and the corresponding XML data values. The XPath expression must evaluate to either (i) a sequence of strings, where each string provides the value of both the visible combo box entry as well as that entry's XML data value (example: `("One", "Two", "Three")`, or (ii) a sequence of two-member arrays, where the first member is the XML data value and the second member is the visible entry  (example: `["One", "1"], ["Two", "2"], ["Three", "3"]`.
- Use two different XPath expressions to generate, respectively, (i) the sequence of strings that are the XML data values, and (ii) the sequence of strings that are the visible entries of the combo box. A one-to-one index mapping between the items of the two sequences determines the correspondence of visible entry to XML value. So, for example, the first items of both sequences are mapped to each other, the second items to each other, and so on.

**Note:** Using XPath expressions enables you to generate entries and XML data values dynamically. For example, the expression `distinct-values(region), "All"` generates a sequence, the items of which are the different distinct values of the **`region`** element, plus a final item, **`All`**.

**Note:** If the items in the drop-down list are obtained from a page source, then they will appear, by default, in document order.

▼ Get Value from XPath

The value returned by the XPath expression is displayed in the control. This enables alternative ways to enter display values for certain controls. For example, a combo box could take its display value from either a page source node or the return value of the **`Get Value from XPath`** property.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the generation of the value of the `Get Value from XPath` property. If used, then a validation error results.

▼ Auto Correct Value

Possible values are `true` or `false`. If set to `true`, then the value in the associated page source node is corrected to the first XML value in the unsorted list of combo box values. The property's default value is `false`.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node 886 action. The `Auto Correct Value` property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

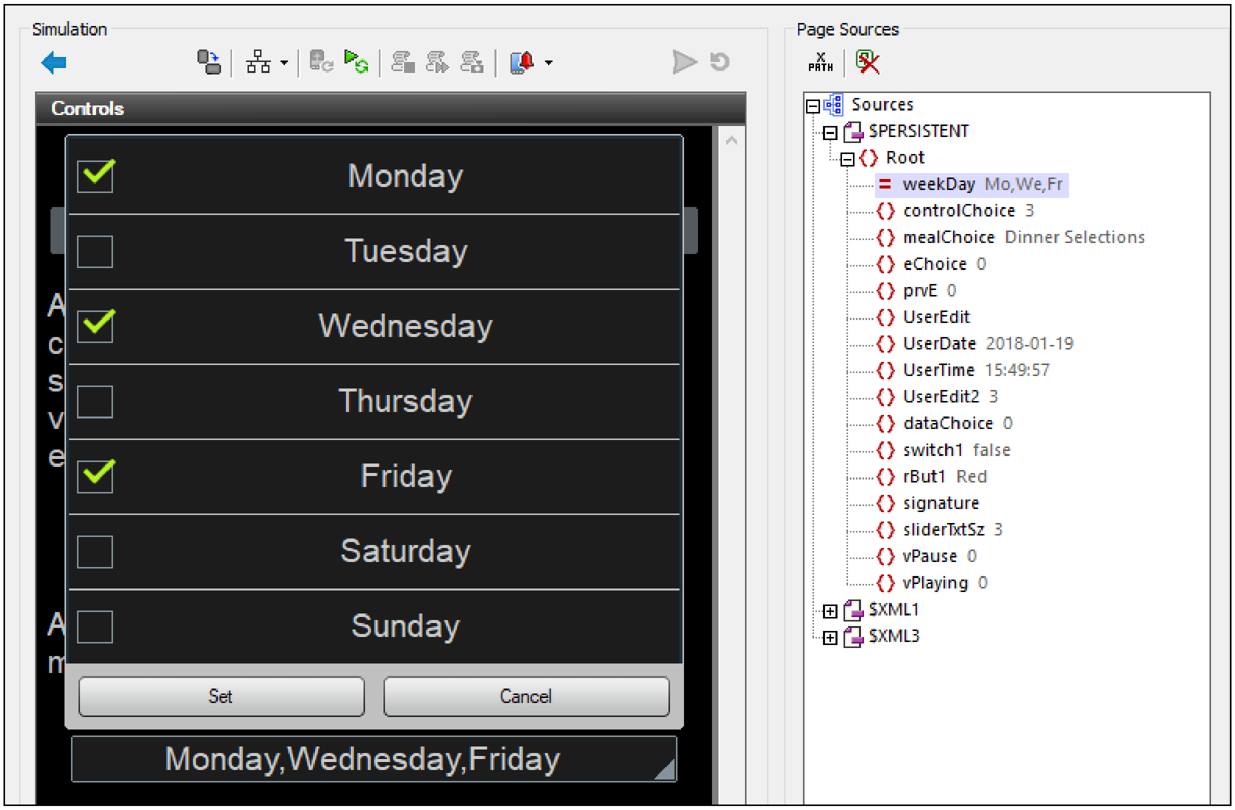
▼ Multi Select

Sets whether the combo box enables multiple values to be selected or not (`true/false`). The default is `false`.

A combo box normally offers a choice of multiple options, of which only one may be selected. In a MobileTogether solution, the value corresponding to the selected option is written to a page source node.

If the `Multi Select` property is set to `true`, then the user may select more than one option. The values corresponding to each of the selected options are concatenated into a single string, and entered in the associated page source node. When the string is constructed, the selected values are separated from each other by a separator that you specify in the **`Multi Select Separator`** property (*see below*). Note that the combo box values themselves must not contain the separator.

In the screenshot below, the values corresponding to the selected options are concatenated into a single string (with comma separators) and entered in the **`weekDay`** attribute node (highlighted). The separator for the string that is entered in the page source is specified via the **`Multi Select Separator`** property (*see below*), while the separator for the string that is displayed in the combo box itself is specified via the **`Multi Select Separator Visible`** property (*see below*).



Sources
SPERSISTENT
Root
weekDay Mo,We,Fr
controlChoice 3
mealChoice Dinner Selections
eChoice 0
prvE 0
UserEdit
UserDate 2018-01-19
UserTime 15:49:57
UserEdit2 3
dataChoice 0
switch1 false
rBut1 Red
signature
sliderTxtSz 3
vPause 0
vPlaying 0
SXML1
SXML3

**Note:** If, in subsequent processing, multiple values contained in a string need to be separated, then the string must be tokenized using the same separator that was used when constructing the string. You can use the XPath **`tokenize`** function for this.

### Multi Select Separator

This property is enabled when the `Multi Select` property is set to `true`. If the end user selects multiple values in the combo box, then a string is constructed from the selections by concatenating the XML values of the selections and inserting the separator between values. The string that is constructed in this way is entered in the associated page source node. Note that a different separator can be used to construct the string that is displayed in the combo box itself; see the `Multi Select Separator Visible` property below. The availability of different separators enables an optimal choice of separator for XML processing on the one hand and a human-readable device display on the other.

You can either: (i) select a separator from the available choices, or (ii) enter one or more characters that you want to use as the separator, either directly or via an XPath expression.

**Note:** If you use an XPath expression to return the separator, note that the expression is evaluated when the combo box values are parsed. So, if the expression is such that its return value changes during runtime processing (for example, if it depends on a node with changing content), then the returned value might not be the desired character/s.

Also see the the `Multi Select` property above for related information.

▼ Multi Select Separator Visible

This property is enabled when the `Multi Select` property is set to `true`. If the end user selects multiple values in the combo box, then a string is constructed from the selections by concatenating the Visible Entry values of the selections and inserting the separator between values. The string that is constructed in this way is displayed in the combo box. Note that a different separator can be used to construct the string that is entered in the associated page source node; see the `Multi Select Separator` property above. The availability of different separators enables an optimal choice of separator for XML processing on the one hand and a human-readable device display on the other.

You can either: (i) select a separator from the available choices, or (ii) enter one or more characters that you want to use as the separator, either directly or via an XPath expression.

**Note:** If you use an XPath expression to return the separator, note that the expression is evaluated when the combo box values are parsed. So, if the expression is such that its return value changes during runtime processing (for example, if it depends on a node with changing content), then the returned value might not be the desired character/s.

Also see the the `Multi Select` property above for related information.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

### Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

### Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

### Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

### Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|`

`large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the [`mt-font-height`](1262) function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is  the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a [control template](1200) to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as

a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to

select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **`XPath`** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

### Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value

directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as high as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as high as the control's content requires.

In effect, `fill_parent` creates a maximum height, while `wrap_content` creates a minimum height.

*Default values*

- For the the Geolocation Map control, the default is the smaller of the two values **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.
- For all other controls, the default is `wrap_content`.

### Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** 1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.6 Date

Date controls are used to format dates obtained from a node in a page source. This is useful, for example, if you wish to format dates differently for different regions: `12-31-2014` (US format) and `31-12-2014` (EU format). The formatting is defined in the `Date/Time Format String` property (*see below for details*). Note that the source node content must be in the correct lexical format as defined by the XSD specification: `YYYY-MM-DD`.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View 253) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the

associated control.

- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog 296).
- A control's CSS properties can be defined in the Styles & Properties Pane 274 and/or in an external CSS file 296. Those defined in the Styles & Properties Pane 274 have priority.

## Date events

The `OnFinishEditing event` 665 is available. For a description of the actions that can be defined for this event, see the Actions section 667.

## Date properties

The control's properties are available in the Styles & Properties Pane 274, and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog 296 has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

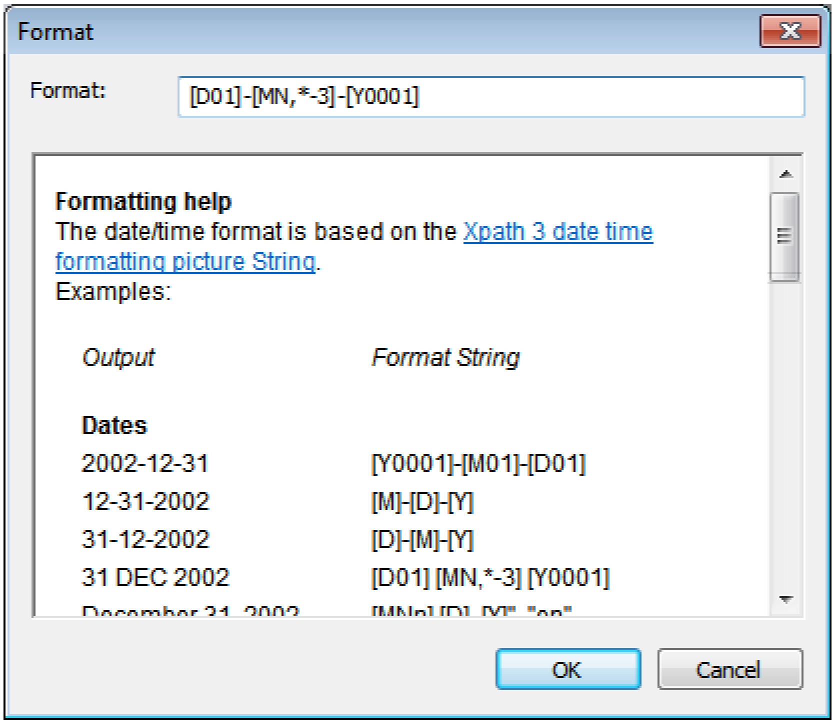
```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
    }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane 274 will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane 274.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Date/Time Format String

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (*screenshot below*).



Format: [D01]-[MN,*-3]-[Y0001]

**Formatting help**
The date/time format is based on the Xpath 3 date time formatting picture String.
Examples:

| *Output* | *Format String* |
|---|---|
| **Dates** | |
| 2002-12-31 | [Y0001]-[M01]-[D01] |
| 12-31-2002 | [M]-[D]-[Y] |
| 31-12-2002 | [D]-[M]-[Y] |
| 31 DEC 2002 | [D01] [MN,*-3] [Y0001] |
| December 31, 2002 | [MNn] [D], [Y]", "en" |

The formatting will be applied to the control's content if the content has the correct lexical form of `xs:date` (for the Date control), `xs:time` (for the Time control), or `xs:dateTime` (for the Date, Time, and DateTime controls). Basic examples are:

- *xs:date:* `2014-12-31`
- *xs:time:* `23:59:59`
- *xs:dateTime:* `2014-12-31T23:59:59`

▼ Date/Time Format Language

Select one of the supported languages from the dropdown list of the combo box (`EN`, `DE`, `ES`, `FR`, `JA`). The selected language will be used in the date/time formatting that is set in the `Date/Time Format String` property (*see description above*). If the names of months and weekdays are used in the format string, then these will be displayed in the language selected for this property. The default language is English.

▼ Time Handling

Specifies how to handle the `Time` part of a value. The three available options are:

- *Cut time*, which removes the `Time` part
- *Set time to zero*, which keeps the `Time` part, but changes it to `00:00:00`
- *Keep time*, which keeps the `Time` part

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Assertion

Sets a condition to be met for the control to be valid. If the assertion is invalid, then the text of the `Assertion Message` property (*see below*) is displayed in the Assertion Message 409 control. (If there are multiple Assertion Message 409 controls, then all these controls will display the text of the `Assertion Message` property.) Click the property's **XPath** icon to enter an XPath expression that defines the assertion. For example: The XPath expression `LastName != ""` asserts that the node `LastName` must not be empty. If this node is empty, then the control's assertion message is displayed in the Assertion Message 409 control of the page.

Note that other controls and the page can also have assertions. If there are multiple invalid assertions on a page, then the assertion message of the first invalid assertion is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

### ▼ Assertion Message

Sets the assertion message to be displayed if the control's assertion is not valid. Double-click inside the value field of the property to edit the assertion message, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. The assertion message is displayed in the Assertion Message 409 control. For example: If the XPath expression of a control's assertion is `LastName != ""`, then it asserts that the node `LastName` must not be empty. If this node is empty, then the assertion message of the control is displayed in the Assertion Message 409 control of the page.

Note that assertions can also be defined for other controls and the page. So it can happen that there are multiple invalid assertions on a page. If this happens, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

### ▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

### ▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal

details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

### Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` 1262 function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

#### ⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is  the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be

automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template 1200 to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal

details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Hint

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Text Color Hint

Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (*see above*). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an

XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`;

the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order**[1633] menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu[404].

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.7 DateTime (iOS)

DateTime controls are available only on iOS client devices and are used to format dateTimes obtained from a node in a page source. This is useful, for example, if you wish to format dateTimes differently for different regions: `12-31-2014 12:00:00` (US format) and `31-12-2014 12:00:00` (EU format). The formatting is defined in the `Date/Time Format String` property (*see below for details*). Note that the source node content must be in the correct lexical format as defined by the XSD specification: `YYYY-MM-DDTHH:MM:SS`. Optional timezone and millisecond parts are allowed.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

## DateTime events

The **`OnFinishEditing` event**[665] is available. For a description of the actions that can be defined for this event, see the Actions section[667].

## DateTime properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### ▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### ▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

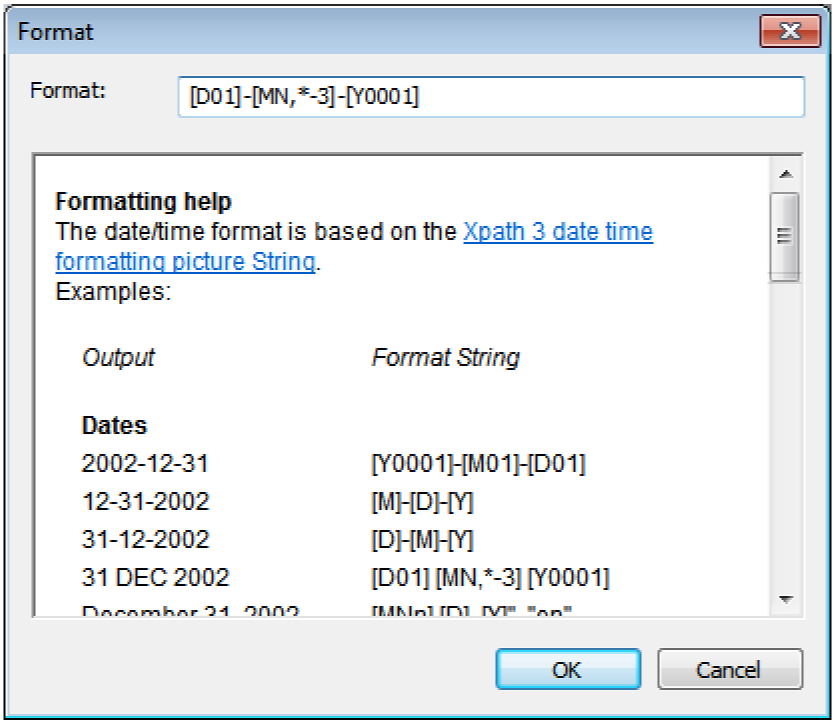
```
map{
    "Bold Text"         : $XML1/R/@bold = "1",
    "Italic Text"       : true(),
    "Text"              : "hello",
    "Text Color"        : "red",
    "Background Color"  : $XML1/R/@background,
    "Text Size"         : $XML1/R/@textsize
   }

map{
    "Style Sheet"       : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Date/Time Format String

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (*screenshot below*).



Format: [D01]-[MN,*-3]-[Y0001]

**Formatting help**
The date/time format is based on the Xpath 3 date time formatting picture String.
Examples:

| *Output* | *Format String* |
|---|---|
| **Dates** | |
| 2002-12-31 | [Y0001]-[M01]-[D01] |
| 12-31-2002 | [M]-[D]-[Y] |
| 31-12-2002 | [D]-[M]-[Y] |
| 31 DEC 2002 | [D01] [MN,*-3] [Y0001] |
| December 31, 2002 | [MNn] [D], [Y], "en" |

The formatting will be applied to the control's content if the content has the correct lexical form of `xs:date` (for the Date control), `xs:time` (for the Time control), or `xs:dateTime` (for the Date, Time, and DateTime controls). Basic examples are:

- *xs:date:* `2014-12-31`
- *xs:time:* `23:59:59`
- *xs:dateTime:* `2014-12-31T23:59:59`

▼ Date/Time Format Language

Select one of the supported languages from the dropdown list of the combo box (`EN`, `DE`, `ES`, `FR`, `JA`). The selected language will be used in the date/time formatting that is set in the `Date/Time Format String` property (*see description above*). If the names of months and weekdays are used in the format string, then these will be displayed in the language selected for this property. The default language is English.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog[667]. You can set actions to perform when a control event[665] is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog[667]. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Assertion

Sets a condition to be met for the control to be valid. If the assertion is invalid, then the text of the `Assertion Message` property (*see below*) is displayed in the Assertion Message[409] control. (If there are multiple Assertion Message[409] controls, then all these controls will display the text of the `Assertion`

`Message` property.) Click the property's **XPath** icon to enter an XPath expression that defines the assertion. For example: The XPath expression `LastName != ""` asserts that the node `LastName` must not be empty. If this node is empty, then the control's assertion message is displayed in the Assertion Message 409 control of the page.

Note that other controls and the page can also have assertions. If there are multiple invalid assertions on a page, then the assertion message of the first invalid assertion is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

▼ Assertion Message

Sets the assertion message to be displayed if the control's assertion is not valid. Double-click inside the value field of the property to edit the assertion message, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. The assertion message is displayed in the Assertion Message 409 control. For example: If the XPath expression of a control's assertion is `LastName != ""`, then it asserts that the node `LastName` must not be empty. If this node is empty, then the assertion message of the control is displayed in the Assertion Message 409 control of the page.

Note that assertions can also be defined for other controls and the page. So it can happen that there are multiple invalid assertions on a page. If this happens, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

### Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height`[1262] function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

#### *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group.

This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.

- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template 1200 to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▾ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▾ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▾ Hint

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▾ Text Color Hint

Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (*see above*). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▾ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### ▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

### ▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu[404].

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.8 Edit Field

Edit fields can be used to allow input from the end user. If the end-user input is intended to be stored in a page source node, then an association must be made in the design with this node (by dropping the node from the page source onto the control). In this case, ensure that the edit field is editable by setting the `Enabled/Editable` property to `true`. The value of the associated node is displayed in the edit field. When the end user modifies the edit field, the associated node is updated automatically.

Instead of associating a node with the edit field, the content of the edit field can be defined by an XPath expression in the `Text` property (*see below*). If the XPath expression locates an XML node, the effect is the same as associating a node by dropping it on to the control. The XPath expression can also be used to calculate an output and display it in the edit field. In this case the `Enabled/Editable` property will be set to `false` automatically, and cannot be edited. It then functions in the same way as a Label control[554].

Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.

- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

## Edit Field events

The **`OnTyping event`**[665] is available. For a description of the actions that can be defined for edit field events, see the Actions section[667].

## Edit Field properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane 274 will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane 274.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Keyboard

Defines the type of keyboard that is displayed on the client when the user starts to edit content. Select one of the following options from the dropdown list of the combo box, or enter an XPath expression to select/generate the value you want. The default is client-specific

- Text
- Number
- Password: *The password is hidden.*
- Visible Password: *The password is shown.*
- Email
- URI
- Phone

▼ Text

The `Text` property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the Rich Text 584 control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text 1227 section for more information.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

▼ Multiline

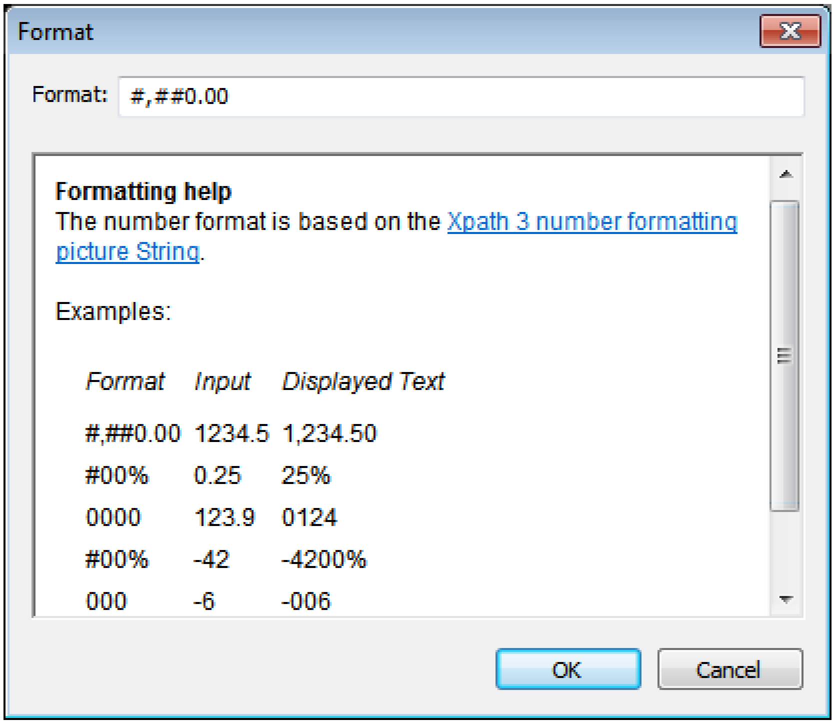
Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

▼ Number Format String

Click the **Additional Dialog** button and enter a number format in the Format dialog that appears (*screenshot below*).



Format: #,##0.00

**Formatting help**
The number format is based on the Xpath 3 number formatting picture String.

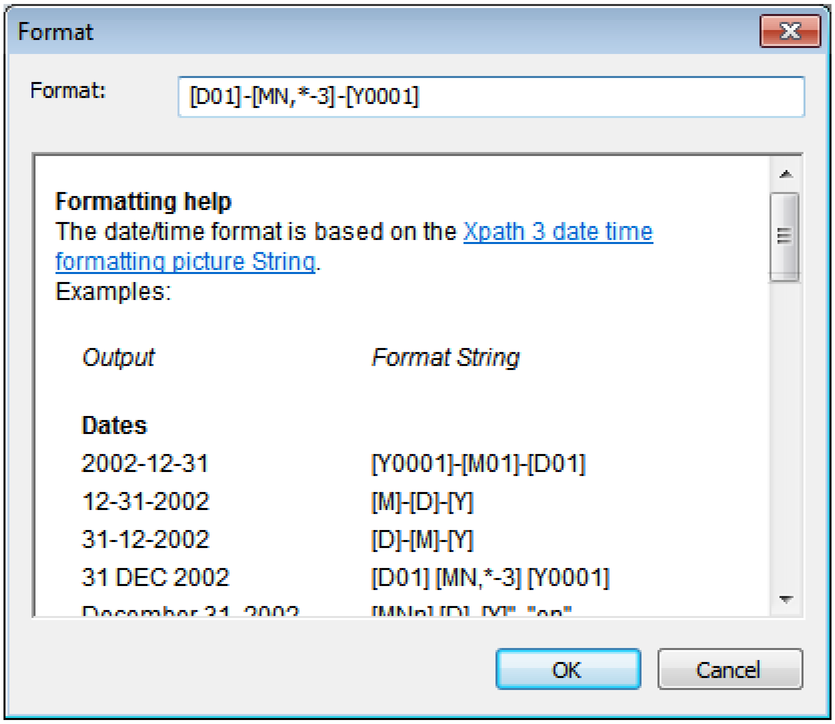
Examples:

| *Format* | *Input* | *Displayed Text* |
|---|---|---|
| #,##0.00 | 1234.5 | 1,234.50 |
| #00% | 0.25 | 25% |
| 0000 | 123.9 | 0124 |
| #00% | -42 | -4200% |
| 000 | -6 | -006 |

The formatting will be applied to the control's content if the content is numeric, and will be displayed in the solution, not in the design.

▼ Date/Time Format String

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (*screenshot below*).



[D01]-[MN,*-3]-[Y0001]

**Formatting help**
The date/time format is based on the Xpath 3 date time formatting picture String.
Examples:

| *Output* | *Format String* |
|---|---|
| **Dates** | |
| 2002-12-31 | [Y0001]-[M01]-[D01] |
| 12-31-2002 | [M]-[D]-[Y] |
| 31-12-2002 | [D]-[M]-[Y] |
| 31 DEC 2002 | [D01] [MN,*-3] [Y0001] |

The formatting will be applied to the control's content if the content has the correct lexical form of `xs:date` (for the Date control), `xs:time` (for the Time control), or `xs:dateTime` (for the Date, Time, and DateTime controls). Basic examples are:

- *xs:date:* `2014-12-31`
- *xs:time:* `23:59:59`
- *xs:dateTime:* `2014-12-31T23:59:59`

Date/Time Format Language

Select one of the supported languages from the dropdown list of the combo box (`EN`, `DE`, `ES`, `FR`, `JA`). The selected language will be used in the date/time formatting that is set in the `Date/Time Format String` property (*see description above*). If the names of months and weekdays are used in the format string, then these will be displayed in the language selected for this property. The default language is English.

Trigger Control Actions

Control actions can be triggered as soon as typing starts, during typing (after a specified time interval), or after typing has been completed. For example, the page can be scrolled to bottom (a control action) when typing starts, or a node can be updated (another control action) after typing has been completed.

- The available values of this property are `OnTyping`, `OnFinishEditing`, or `AfterTimeInterval`.
- The default value is `OnFinishEditing` for web clients and `OnTyping` for all other clients.
- You can select the value you want or enter it via an XPath expression (by clicking the XPath toolbar icon).
- If `AfterTimeInterval` is selected, then the **`Trigger Control Actions on Typing Interval [ms]`** property becomes available *(see below)*.
- Editing is considered to be finished when **Return** is pressed or when the end user taps outside

the edit field.

▼ Trigger Control Actions on Typing Interval [ms]

This property becomes available if the **`Trigger Control Actions`** property is set to a value of `AfterTimeInterval` *(see above)*. It specifies the time interval in milliseconds from when typing starts to when the control action/s is/are triggered. The default interval is 500ms.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Assertion

Sets a condition to be met for the control to be valid. If the assertion is invalid, then the text of the `Assertion Message` property *(see below)* is displayed in the Assertion Message 409 control. (If there are multiple Assertion Message 409 controls, then all these controls will display the text of the `Assertion Message` property.) Click the property's **XPath** icon to enter an XPath expression that defines the assertion. For example: The XPath expression `LastName != ""` asserts that the node `LastName` must not be empty. If this node is empty, then the control's assertion message is displayed in the Assertion

Message[409] control of the page.

Note that other controls and the page can also have assertions. If there are multiple invalid assertions on a page, then the assertion message of the first invalid assertion is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

▼ Assertion Message

Sets the assertion message to be displayed if the control's assertion is not valid. Double-click inside the value field of the property to edit the assertion message, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. The assertion message is displayed in the Assertion Message[409] control. For example: If the XPath expression of a control's assertion is `LastName != ""`, then it asserts that the node `LastName` must not be empty. If this node is empty, then the assertion message of the control is displayed in the Assertion Message[409] control of the page.

Note that assertions can also be defined for other controls and the page. So it can happen that there are multiple invalid assertions on a page. If this happens, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

### Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` [1262] function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* [1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** [1304] and **`$MT_CanvasY`** [1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

### Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

### Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

### Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to

select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Hint

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Text Color Hint

Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (*see above*). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-

independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows:

The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order**[1633] menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu[404].

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.9 Geolocation Map

A Geolocation Map control can be used to display the map of a specific area: in street, satellite, or hybrid view (the view is defined in the `Map Type` property). Points of interest in the area covered by the map can be shown by markers. Any number of markers can be defined in the control's **`Markers`** property. The current location as well as zoom controls can optionally be included in the map (see their respective properties below). The size and magnification of the map that is initially displayed in the control can be defined via the `Viewport`, `Control Width`, and `Control Height` properties. The Geolocation Map control also has a `OnGeoMapMarkerClicked`[508] event, which enables you to define actions to perform when a particular marker is clicked by the user.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

### Geolocation Map events

The **`OnGeoMapMarkerClicked event`**[665] is available, which enables you to define actions to perform when a marker in the map is clicked. In order to retrieve information about the marker that has been clicked, use the **`$MT_GeolocationMapMarker`**[1304] dynamic variable.

The dynamic variable **$MT_GeolocationMapMarker** contains information about the marker that was last clicked by the client's user. This information that is contained in the variable is stored in an XPath-map construct, in a format that is shown by the following example:

```
map {
     "id":"vie",
     "geolocation":(48.2143531, 16.3707266),
     "title":"Vienna",
     "text":"Altova EU"
}
```

To fetch a value from the XPath-map construct, use an XPath expression like this: `map:get( $MT_GeolocationMapMarker, "id" )`. This particular expression returns the value of the **id** key (that is, the **id** of the marker that was clicked).

For a description of the actions that can be defined for the **OnGeoMapMarkerClicked** event, see the Actions section[667].

## Geolocation Map properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling

properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Map Type

Specifies the map type that is displayed in the Geolocation Map control. The options are:

- *Default (no selection):* Uses the map type that is selected on the client device
- *Street:* Shows a street plan layout of the displayed
- *Satellite:* Shows a satellite picture of the displayed area
- *Hybrid:* Shows a combination of the street and satellite map types

**Note:** An XPath expression can also be used to specify the map type dynamically. The expression must evaluate to one of the following: (i) **`street`**, (ii) **`satellite`**, (iii) **`hybrid`**.

**Note:** On web clients and in the simulator, only a single map type is available no matter which option from those listed above is selected.

### Show Current Location

Specifies whether the current location of the client device is displayed in the map. This would be useful to help users locate themselves in the map. The values of the property are `true` or `false`, with the default being `false`. To define the property's value dynamically, use an XPath expression (for example: **`Element/@attribute="1"`** would evaluate to `true` or `false` depending on the value of **`Element/@attribute`**).

**Note:** For this feature to work, GPS permission must be granted on the client device.

### Show Zoom Controls

Specifies whether zoom controls are displayed in the map. The values of the property are `true` or `false`, with the default being `true`. You can use an XPath expression to define the property's value dynamically. Note that zoom controls are not displayed on iOS devices.

### Markers

An XPath expression that defines the number of markers that are placed in the map, as well as their properties. The XPath expression must resolve to one or more XPath-map constructs: **`map-1, map-2, ... map-N`**. Each XPath-map construct consists of a set of key–value pairs that together define the properties of one marker. The following keys are specified:

- **`id:`** The value is an ID for the marker
- **`geolocation:`** This is a sequence of two strings or two numbers (integer or decimal): the latitude and the longitude; for example: `(48.2143531, 16.3707266)`
- **`title:`** A string that is the title of the popup that appears when the marker (in the map) is clicked
- **`text:`** A string that is the text of the popup that appears when the marker (in the map) is clicked

A full XPath-map construct would look something like this:

```
map {
     "id":"vie",
     "geolocation":(48.2143531, 16.3707266),
     "title":"Vienna",
     "text":"Altova EU"
}
```

The XPath-map construct above would create a marker for the Altova EU office in Vienna. To add more markers to the map, enter additional XPath-map constructs with a preceding comma separator: **`map-1, map-2, ... map-N`**.

*mt-geo-map-marker()*
The MobileTogether XPath extension function `mt-geo-map-marker` can be used to generate a marker dynamically. Each **`mt-geo-map-marker`** function returns an XPath-map construct, that is, one marker. To create multiple markers, multiple **`mt-geo-map-marker`** functions can be used. For more information about the function, see the description of **`mt-geo-map-marker()`** 1262.

- Viewport

An XPath expression that evaluates to two geolocation coordinates: the top-left corner and the bottom-right corner of the viewport rectangle. The viewport is the geolocation area that is shown in the geolocation map. Note, however, that the dimensions of the Geolocation Map control are defined by the Control Width and Control Height properties. The viewport, if correctly defined, will lie within the map area shown in the control.

The XPath expression must evaluate to a sequence of two strings, where each string is one set of coordinates. In the following example, for instance, the first string is the top-left coordinate while the second string is the bottom-right coordinate: `("61° -10°", "41° 10°")`. The top-left coordinate would thus be 61°N 10°W, while the bottom-right coordinate would be 41°N 10°E. These coordinates would create a viewport of roughly 10° square centered on London (approx. 51°N 0°E).

The default setting for the viewport are the coordinates of a rectangle that displays all defined markers at the highest possible zoom level.

*geolocations-bounding-rectangle()*
The Altova extension function **`geolocations-bounding-rectangle`** 1707 can be used to create a bounding rectangle around a set of submitted geolocations. The function returns the top-left corner and bottom-right corner of a rectangle that fits around the submitted geolocations at the greatest possible zoom level, that is, as closely as possible around the geolocations. The return value of the function is a sequence of two strings; so the function can be used in an XPath function to generate the value of the Viewport property.

For more information about the function, see the description of **`geolocations-bounding-rectangle()`** 1707
.

- Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value

is selected, the property `Max Control Width` becomes available

- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as high as the parent, which could be, for example, a table cell

or the page
- `wrap_content`: makes the control only as high as the control's content requires.

In effect, `fill_parent` creates a maximum height, while `wrap_content` creates a minimum height.

*Default values*
- For the the Geolocation Map control, the default is the smaller of the two values **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.
- For all other controls, the default is `wrap_content`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu[404].

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.10 Horizontal Line

The Horizontal Line control enables you to add lines that you can style for color and width.

Notes

- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

## Horizontal Line events

There is no event associated with the Horizontal Line control.

## Horizontal Line properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Line Width

Sets the width (thickness) in pixels of the line. Select a width value (in pixels, dp, or sp) from the dropdown list of the combo box, or double-click in the value field to enter a numeric value. You can also use an XPath expression. The numeric value is understood to be a pixel value. For this reason, no unit should be specified.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Line Style

Specifies the style of the line. You can select one of the options from the dropdown list of the combo box, or use an XPath expression. The default value is `solid`.

▼ Line Color

Specifies the color of the line. You can do one of the following to select the color:

- Click the color palette to select a line color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate or fetch the required color code

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▾ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▾ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▾ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.11 Horizontal Slider

The Horizontal Slider control enables users to select a value along the slider's scale. The selected value can be entered as the value of a page source node (via a page source link; *see Notes below*). In the Styles & Properties Pane 274, you can set the slider's minimum and maximum values (*see 'Horizontal Slider properties' below*), and specify control actions 667 to execute, either when the slider is being moved or when it has finished being moved.

The three ways in which control actions can be executed (based on the sliding action) are described below:

| **Action execution** | **`Trigger Control Actions During Sliding`** | **`Slider's Trigger Interval (in ms)`** |
|---|---|---|
| *Actions triggered after sliding completed* | `false` (default) | — |
| *Actions triggered continuously* | `true` | `0` |
| *Actions triggered at interval* | `true` | `1 – 10000` |

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View 253) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog 296).
- A control's CSS properties can be defined in the Styles & Properties Pane 274 and/or in an external CSS file 296. Those defined in the Styles & Properties Pane 274 have priority.

## Horizontal Slider events

The **`OnSliding event`** 665 is available. For a description of the actions that can be defined for Horizontal Slider events, see the Actions section 667.

## Horizontal Slider properties

The control's properties are available in the Styles & Properties Pane 274, and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Get Value from XPath

The value returned by the XPath expression is displayed in the control. This enables alternative ways to enter display values for certain controls. For example, a combo box could take its display value from either a page source node or the return value of the **`Get Value from XPath`** property.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the generation of the value of the `Get Value from XPath` property. If used, then a validation error results.

### Auto Correct Value

Possible values are `true` or `false`. The property's default value is `false`. If set to `true` and the entered value is outside the slider's defined range of values, then the entered value is automatically corrected to a value from the range that is closest to the entered value. (The slider's defined range of values is set with the `Slider Minimum Value` and `Slider Maximum Value` properties.) For example, if the property's value is set to `true`, the slider's range of values is defined to be between `0` and `100` (the default range), and the slider's associated page source node contains a value of `"somestring"`, then this value is automatically corrected to `0`. If the node contained a value of `200`, then its value would be auto-corrected to `100`. The auto-correction happens immediately the out-of-range value is detected, for example, when the solution is opened or when the value is entered.

▼ Slider Minimum Value

Sets the minimum value of the slider. Allowed values are -10000 to 10000. The default value is 0. Select one of the predefined numbers between 0 and 100, or enter an XPath expression.

▼ Slider Maximum Value

Sets the maximum value of the slider. Allowed values are -10000 to 10000. The default value is 100. Select one of the predefined numbers between 0 and 100, or enter an XPath expression.

▼ Slider Color

The Slider Color property sets the color of the slider's scale line when the slider is enabled. The default value is client-specific.

▼ Slider Color (Disabled)

The Slider Color (Disabled) property sets the color of the slider's scale line and the slider's marker when the slider is disabled. The condition/s for enabling and disabling a slider is set on the slider's Enabled/Editable property. The property's default value is client-specific.

▼ Slider Thumb Color

The Slider Thumb Color property sets the color of the slider's marker when the slider is enabled. The default value is client-specific.

▼ Trigger Control Actions During Sliding

If set to OnSliding, control actions are triggered **during** sliding (either continuously or at specified intervals). When set to OnSliding, the property Trigger Control Actions on Sliding Interval [ms] *(see below)* appears; you can set the trigger intervals here. If set to OnFinishEditing, control actions are triggered **after sliding has finished**. Default is OnFinishEditing.

▼ Trigger Control Actions on Sliding Interval (ms)

Specifies the intervals, in milliseconds, at which control actions are triggered. The range of values is 0 to 10000. The default is 1000ms. Select a value from the dropdown list, or enter a value.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

### Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

### Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

### Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal

details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**[1304] and **$MT_CanvasY**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

## ▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

## ▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** 1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

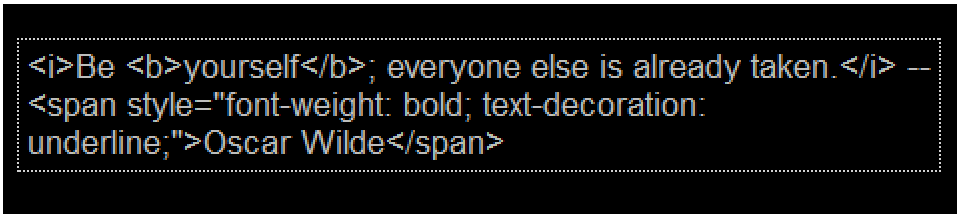
You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.12 HTML Label

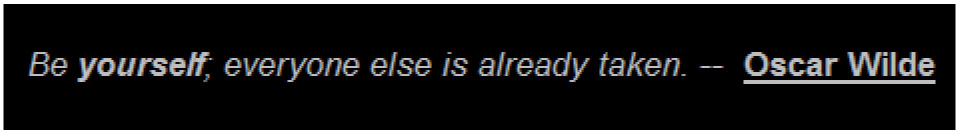
HTML Labels can be used if you want to to generate varying styles within a single label.

When you use a regular Label control[554], the styles that you assign in the Styles & Properties pane are assigned to the entire label and the text within the label cannot have different styles. Inside an HTML label, however, you can enter text that is marked up with HTML elements/attributes and CSS styles. The screenshot below shows an HTML Label in the design. With the **`Multiline`** property set to `true`, the text will run on across lines and will not be truncated by the control's width.



```
<i>Be <b>yourself</b>; everyone else is already taken.</i> --
<span style="font-weight: bold; text-decoration:
underline;">Oscar Wilde</span>
```

When this text is displayed on a mobile device, it will be displayed with the formatting defined by the HTML markup you entered.



*Be **yourself**; everyone else is already taken.* -- **Oscar Wilde**

*HTML elements*
In an HTML Label, you can use the following HTML elements (marked red below):

```
<a href="https://www.altova.com">Creates a link</a>
<b>bold</b>, <strong>bold</strong>
<i>italic</i>, <cite>italic</cite>, <dfn>italic</dfn>, <em>italic</em>
<u>underline</u>
<strike>line-through</strike>
```

*HTML attributes*
You can use **`href`** and **`style`** attributes, marked green below:

```
<a href="...">Creates a link</a>
<span style="font-style: italic; font-weight: bold">bold and italic</span>
```

*CSS styles*
You can use the CSS style properties **`font-style`** , **`font-weight`**, and **`text-decoration`**, marked blue below:

```
<span style="font-style: italic; font-weight: bold">bold and italic</span>
```

*Block elements*
HTML block elements are not supported. To create a block, in an HTML Label, use a newline before and a newline after the text you want to mark as a block.

*Marked-up-text only*

The text content of the HTML Label must be directly entered as HTML-marked-up text. In this sense, the text is static content. You can enter the marked-up text directly or generate it dynamically via an XPath expression. You cannot, however, enter dynamic content as part of the static content; if you do, then the entire static content will be replaced by the dynamic content and you will need to format it via the Styles & Properties pane.

⊟ Notes

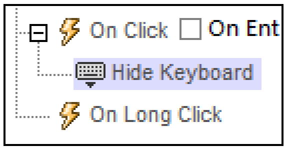
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

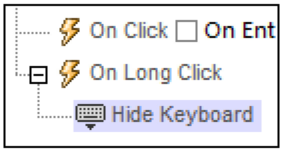
## HTML Label events

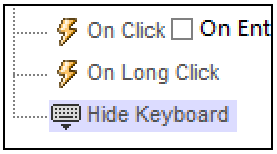
The **`OnLabelClicked`** event is available. To define actions for the label's **`OnLabelClicked`** event, right-click the label and, from the context menu that appears, select **Control Actions for OnLabelClicked**. This displays the Actions dialog for label events. For a description of the actions that can be defined for this event, see the Actions section[667].

▼ OnLabelClicked (OnClick, OnLongClick)

The end user can click the control in one of two ways: a short tap (`On Click`) or a longer press (`On Long Click`). A sequence of different actions[667] can be specified for each type of click (*see screenshots at left and middle below*). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions[667] be executed after those of the end-user click; these actions are defined after the `On Long Click` event (*see screenshot below right*)..

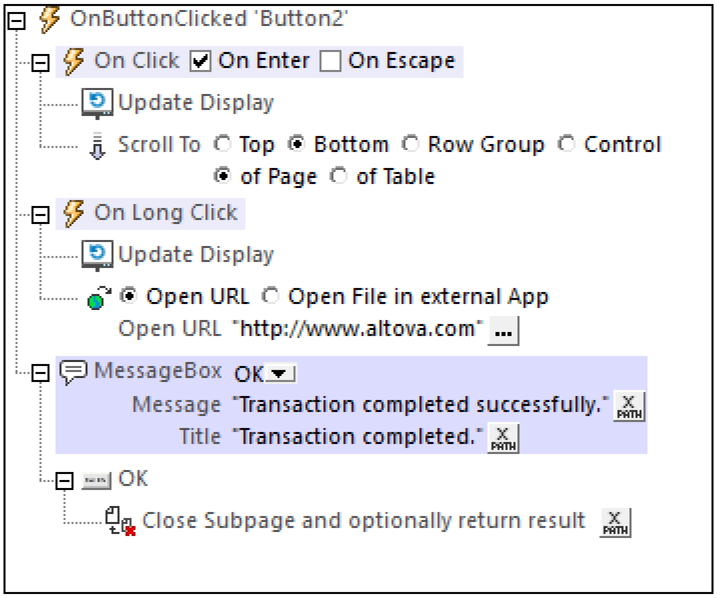






- **`On Click`**: The action/s to perform when the control is tapped (*see screenshot above left*).
- **`On Long Click`**: The action/s to perform when the control is pressed for a longer time than a tap (*see screenshot above center*).
- *Additional actions:* The action/s to perform after the `On Click` or `On Long Click` actions have been executed (*see screenshot above right*). If no action has been set for the `On Click` or `On Long Click` events, then the additional action/s are performed directly on a click or long-click.

You can combine actions[667] for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.



OnButtonClicked 'Button2'

On Click ☑ On Enter □ On Escape

Update Display

Scroll To ○ Top ● Bottom ○ Row Group ○ Control
● of Page ○ of Table

On Long Click

Update Display

● Open URL ○ Open File in external App
Open URL "http://www.altova.com" ...

MessageBox OK
Message "Transaction completed successfully."
Title "Transaction completed."

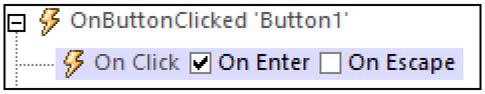
OK

Close Subpage and optionally return result

The screenshot above shows that the `On Click` and `On Long Click` events each has a sequence of actions defined for it. An additional `MessageBox` event is defined after the `On Long Click` event. This `MessageBox` event will be executed after the `On Click` or `On Long Click` sequence of actions has completed.

*On Enter/Escape*

If the control's *On Enter* or *On Escape* check box is selected, then the control's actions are executed when the respective key (**Enter** or **Escape**) is tapped. The key-tap (**Enter** or **Escape**) serves as an alternative to the `On Click` event, and will work additionally to the click. The screenshot below shows the *On Enter* and *On Escape* check boxes of the Button event. Other controls that provide this option look similar and work similarly.



OnButtonClicked 'Button1'

On Click ☑ On Enter □ On Escape

This setting can also be accessed via the control's `On Enter/Escape` property, which is described below.

**Note:** If you select the **Page | Show/Define Tab Order**1633 menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

## HTML Label properties

The control's properties are available in the Styles & Properties Pane274, and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog [296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane [274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane [274].
- You can also specify a style sheet to use, as shown in the second map above.

### HTML Text

The `HTML Text` property takes as its value a fixed value text string. If the text string of the HTML label is marked up with HTML tags, then the text is displayed on client devices with the formatting defined by the HTML markup.

Double-click inside the value field to edit. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

### Multiline

Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes

available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

### Max Number of Lines

This property become available only when the `Multiline` property has been set to `true`. It sets the maximum number of lines that are allowed in the control. The property's default value is `unlimited`. This property cannot be set if the `Text Size Auto-Fit` property has been enabled. You must choose either to set a maximum number of lines or to auto-fit the text.

### Control Action

Click the **Additional Dialog** button to display the control's Actions dialog[667]. You can set actions to perform when a control event[665] is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog[667]. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

### Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

### Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box

- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

▼ Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` 1262 function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template[1200] to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

### Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

### Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

### Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Strikethrough Text

Select `true` or `false` from the dropdown list of the combo box to determine whether a line is drawn through the text of the control. The value can also be generated dynamically via an XPath expression. The default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

## ▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

## ▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the

resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ On Enter/Escape

Takes one of three values:

- `OnEnter`: Specifies that the actions of this control are executed when the **Enter** key is tapped.
- `OnEscape`: Specifies that the actions of this control are executed when the **Escape** key is tapped.
- None: No action when either **Enter** or **Escape** is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to **`"OnEnter"`** or **`"OnEscape"`**. If more than one control on a page is given the same value (`OnEnter` or `OnEscape`), then the first visible and enabled control that has the value is selected when the key is tapped. (See the **`Visible`** and **`Enabled/Editable`** properties.)

This setting can also be made via the dialog to set the control's **`OnClicked`** actions (see the description of the control's events above).

**Note:** If you select the **Page | Show/Define Tab Order**1633 menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu404.

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*1327). See the section Style Sheets1318 for more information.

Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.13 Image

The Image control inserts an image in the design. The image that is selected for insertion can be an image file referenced by a URL, or it can be a string that is Base64-encoded image data. The `Image Source Type` property specifies which of the two types the image is: a URL-located file, or a Base64 string. To specify that the image (URL or Base64 string) is taken from a page source node, drop that node onto the image control. The Image control's properties are listed below.

**Note:** If the image source (URL or Base64 string) is changed during simulation or while the solution runs, then the image must be explicitly reloaded with the Reload action[801]. For example, if a combo box selection changes the selection of an image, a Reload action[801] targeting the image must be defined on the combo box.

⊟ Notes

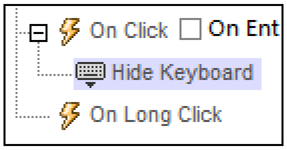
- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

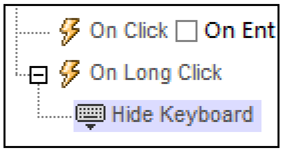
### Image events

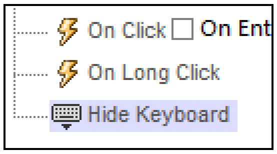
The **`OnImageClicked`** event is available. To define actions for the image's **`OnImageClicked`** event, right-click the image and, from the context menu that appears, select **Control Actions for OnImageClicked**. This displays the Actions dialog for image events. For a description of the actions that can be defined for this event, see the Actions section 667.

OnImageClicked (OnClick, OnLongClick)

The end user can click the control in one of two ways: a short tap (`On Click`) or a longer press (`On Long Click`). A sequence of different actions 667 can be specified for each type of click (*see screenshots at left and middle below*). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions 667 be executed after those of the end-user click; these actions are defined after the `On Long Click` event (*see screenshot below right*)..

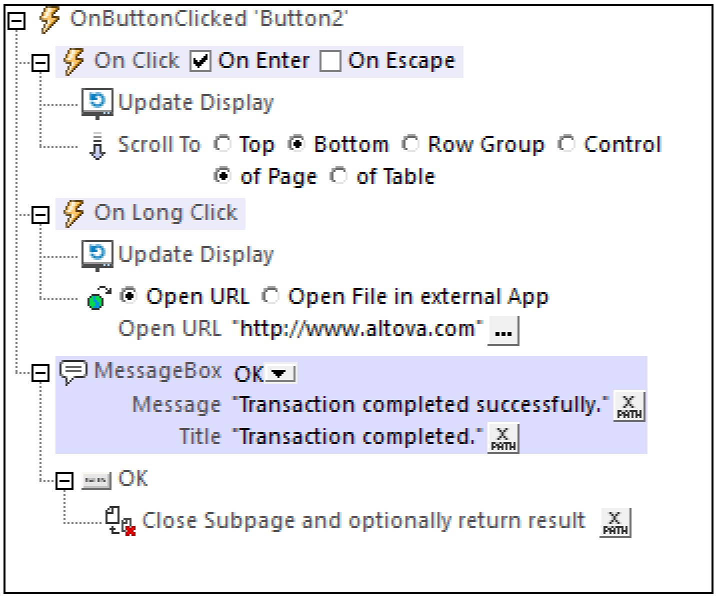






- **`On Click`**: The action/s to perform when the control is tapped (*see screenshot above left*).
- **`On Long Click`**: The action/s to perform when the control is pressed for a longer time than a tap (*see screenshot above center*).
- *Additional actions:* The action/s to perform after the `On Click` or `On Long Click` actions have been executed (*see screenshot above right*). If no action has been set for the `On Click` or `On Long Click` events, then the additional action/s are performed directly on a click or long-click.

You can combine actions 667 for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.

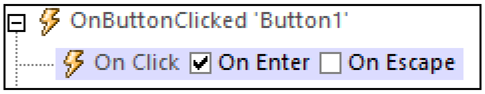


OnButtonClicked 'Button2'
On Click ☑ On Enter ☐ On Escape
Update Display
Scroll To ○ Top ● Bottom ○ Row Group ○ Control
● of Page ○ of Table
On Long Click
Update Display
● Open URL ○ Open File in external App
Open URL "http://www.altova.com" ...
MessageBox OK
Message "Transaction completed successfully."
Title "Transaction completed."
OK
Close Subpage and optionally return result

The screenshot above shows that the `On Click` and `On Long Click` events each has a sequence of actions defined for it. An additional `MessageBox` event is defined after the `On Long Click` event. This `MessageBox` event will be executed after the `On Click` or `On Long Click` sequence of actions has completed.

*On Enter/Escape*

If the control's *On Enter* or *On Escape* check box is selected, then the control's actions are executed when the respective key (**Enter** or **Escape**) is tapped. The key-tap (**Enter** or **Escape**) serves as an alternative to the `On Click` event, and will work additionally to the click. The screenshot below shows the *On Enter* and *On Escape* check boxes of the Button event. Other controls that provide this option look similar and work similarly.



OnButtonClicked 'Button1'

On Click ☑ On Enter □ On Escape

This setting can also be accessed via the control's `On Enter/Escape` property, which is described below.

**Note:** If you select the **Page | Show/Define Tab Order**[1633] menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

## Image properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both

syntax and values.

- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Image Source

The value of the `Image Source` property references an image in one of the following ways:

- The URL of a binary image file (`PNG`, `BMP`, etc). The value of the property must be a URL. The URL is selected in the Specify File dialog (*see description below*).
- An image file represented as a Base64-encoded string. The value of the property must be a Base64-encoded string. An XPath expression supplies the string, which could be entered directly or obtained form an XML node.
- An SQL SELECT statement that queries a page source on the server. The query should return the Base64-encoded string that is to be used as the image. The SELECT statement is generated by an XPath expression.

The type of image source is provided by the property `Image Source Type` (*see next property below*). By default, `Image Source Type` is set to `url`. The `Image Source` property automatically opens the corresponding dialog: Specify File dialog for `url` *(see below)*, and Edit XPath XQuery Expression dialog[1244] for `base64` (*see Base64-Encoded Images*[1098]).

**Note:** If the image source is a URL and the URL is changed during simulation or while the solution runs, then the image must be explicitly reloaded with the Reload action[801]. For example, if a combo box selection changes the selection of an image, a Reload action[801] targeting the image must be defined on the combo box.

⊟ *The Specify File dialog*

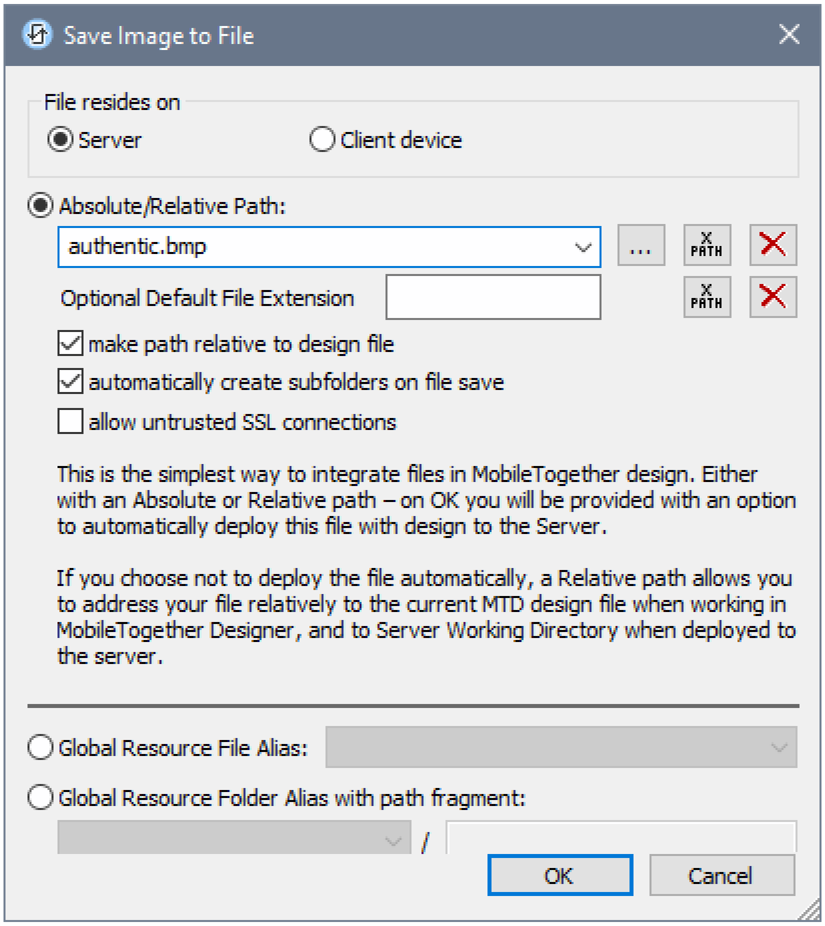
You can choose a file on the server or the client. Select the respective radio button option and, as required, enter details and select options (*see below*).

After finishing and clicking **OK**, a dialog appears asking whether you want to deploy the file to the server and/or client and whether you want to embed the image in the design file.

- *Deploy to Server:* The fie will be downloaded from server to client when the solution is run. This could add to the time-overhead when running the solution.
- *Deploy to Client:* The file is deployed to the client at the time the solution is downloaded.
- *Embed Image in Design File:* The image is embedded in the design file (in Base64 format). This option is useful when you want to share the design file without having to share the image file and the folder structure that contains the file. Selecting this automatically set the **`Embed Image`**[541] property to `true`. Note, however, that for deployment, you still need to specify where the file is to be deployed (server and/or client).

*File is located on server*

If the image file is located on the server, you can either browse for it (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want (*see* *screenshot below*).



Save Image to File

File resides on

● Server ○ Client device

● Absolute/Relative Path:

authentic.bmp

Optional Default File Extension

☑ make path relative to design file

☑ automatically create subfolders on file save

□ allow untrusted SSL connections

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:

○ Global Resource Folder Alias with path fragment:

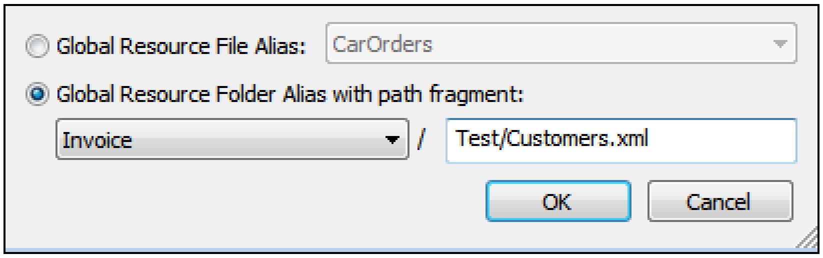
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). See the section Altova Global Resources 1334 for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

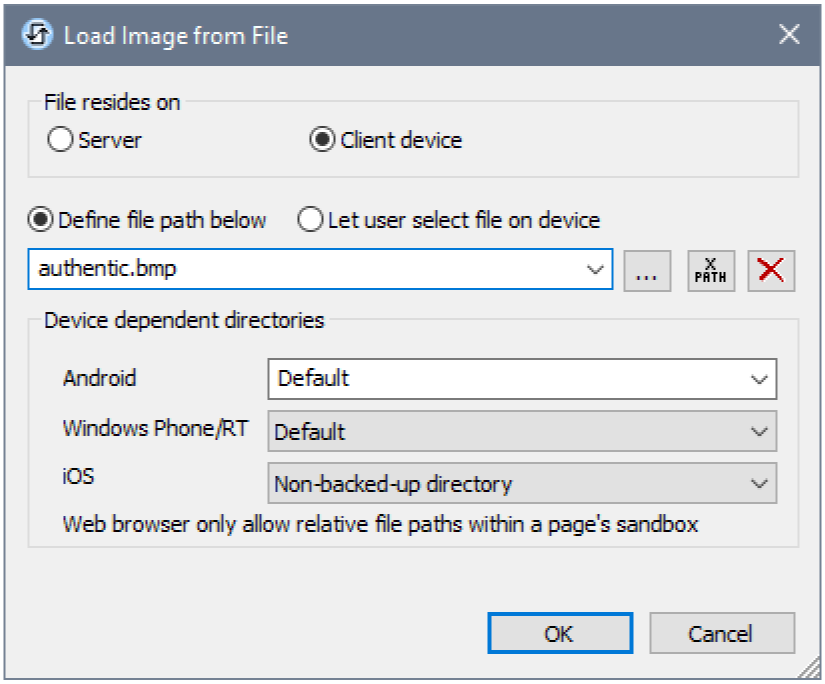
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 1334 for details.

*File is located on client*

If the image file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



File resides on

○ Server ● Client device

● Define file path below ○ Let user select file on device

authentic.bmp

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To[686] (*Source File* and *Target File* options), Load/Save File[809], Load/Save Image[707], Load/Save Binary File[815], Load/Save Text File[821], Read Folder[847], and Get File Info[849].

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.
- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for

files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog 1663 (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html','xml'`).
- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.
- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

▼ Image Source Type

Sets the type of the image source selected by the `Image Source` property above. Three type options are available:

- `url`: a binary image file, such as a `PNG` or `BMP` image file; this is the default image source type
- `base64`: a Base64-encoded string to be used as the image
- `SQL`: an SQL SELECT statement that queries the page source on the server and returns a Base64-encoded string to be used as the image

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is a URL, a Base64 string, or an SQL SELECT statement.

▼ Username

This property is enabled if `Image Source Type` is `url`. Sets a user name for user access to the resource. Double-click in the property's value field to edit.

▼ Password

This property is enabled if `Image Source Type` is `url`. Sets a password for user access to the resource. Double-click in the property's value field to edit.

▼ Create Before Load

In the combo box, select the value you want: `true` or `false`. If `true`, the chart or Base64 image is created before the page loads. If `false`, a page sources action must be used to create the chart or image. The default value is `true`.

▼ Autorotate

Specifies whether to rotate the image according to its EXIF orientation information. Values are **`true`** or **`false`**, with the default being **`false`**.

▼ Embed Image

This property is enabled if `Image Source Type` is `url` (which is the default value). It takes either `true` or `false`. If `true`, then the image is embedded in the design file. The binary data of the image file (`PNG`, `BMP`, etc) is converted into text-based Base64-encoding. This text is embedded in the design file. The default value of the property is `false` (the file is not converted and not embedded).

This property can be automatically set to `true` if, when selecting an image, the option to embed the image is checked. See the **`Image Source`** 541 property.

Once an image is embedded in this way, two useful design-side actions are available:

- If the original image has been modified, you can re-embed it by right-clicking the Image control and select **Deploy/Embed File | Re-embed Image**.
- If you want to save the embedded image to file, right-click the Image control and select **Deploy/Embed File | Save Embedded Image**.

**Note:** The value of a `url` source type must be a URL. If it is an XPath expression that calculates the URL, then the **`Embed Image`** property will not be enabled.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's

**Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

### Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max`

`Control Width` becomes available

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as high as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as high as the control's content requires.

In effect, `fill_parent` creates a maximum height, while `wrap_content` creates a minimum height.

*Default values*

- For the the Geolocation Map control, the default is the smaller of the two values **`$MT_CanvasX`** [1304] and **`$MT_CanvasY`** [1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** [1304] and **`$MT_CanvasY`** [1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.
- For all other controls, the default is `wrap_content`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Max Control Height

The `Max Control Height` property sets the maximum height of the control. Select a value from the property's combo box or enter a value directly. The height may be given in any of the following units: *pixel, dp, or sp*. The value must include the length unit, similarly to the entries in the combo box.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* [1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** [1304] and **`$MT_CanvasY`** [1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### On Enter/Escape

Takes one of three values:

- `OnEnter`: Specifies that the actions of this control are executed when the **Enter** key is tapped.
- `OnEscape`: Specifies that the actions of this control are executed when the **Escape** key is tapped.
- None: No action when either **Enter** or **Escape** is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to **`"OnEnter"`** or **`"OnEscape"`**. If more than one control on a page is given the same value (`OnEnter` or `OnEscape`), then the first visible and enabled control that has the value is selected when the key is tapped. (See the **`Visible`** and **`Enabled/Editable`** properties.)

This setting can also be made via the dialog to set the control's **`OnClicked`** actions (see the description of the control's events above).

**Note:** If you select the **Page | Show/Define Tab Order** 1633 menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

### Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

### Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.14 Label

Labels are used to display information. The information displayed could be static, entered by the designer as the `Text` property of the label. Or it could be dynamic, obtained at runtime from a node in one of the page sources; this dynamic link is known as a page source link. To specify dynamic content for the label (a page source link), either drag the required data tree node from the Page Sources Pane onto the label, or enter, as the `Text` property of the label, an XPath expression that retrieves data from a node or calculates an output.

⊟ Notes

- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

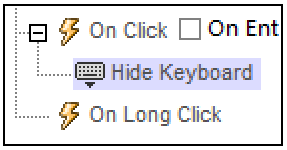
### Label events

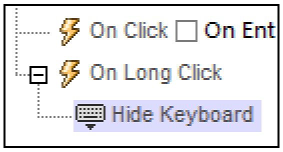
The **`OnLabelClicked`** event is available. To define actions for the label's **`OnLabelClicked`** event, right-click the label and, from the context menu that appears, select **Control Actions for OnLabelClicked**. This displays the Actions dialog for label events. For a description of the actions that can be defined for this event, see the Actions section 667.

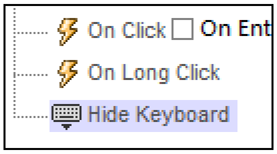
▼ OnLabelClicked (OnClick, OnLongClick)

The end user can click the control in one of two ways: a short tap (`On Click`) or a longer press (`On Long Click`). A sequence of different actions 667 can be specified for each type of click (*see screenshots at left and middle below*). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions 667 be executed after those of the end-user click;

these actions are defined after the `On Long Click` event (*see screenshot below right*)..

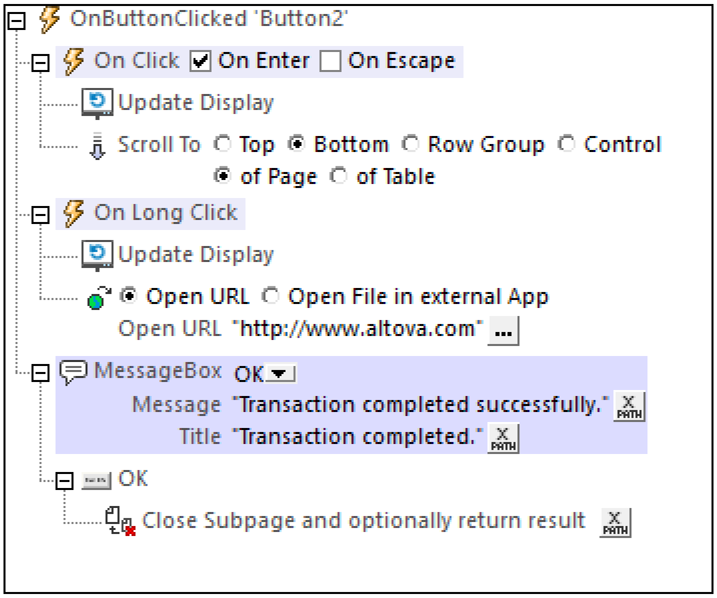

On Click □ On Ent
Hide Keyboard
On Long Click


On Click □ On Ent
On Long Click
Hide Keyboard


On Click □ On Ent
On Long Click
Hide Keyboard

- **`On Click`**: The action/s to perform when the control is tapped (*see screenshot above left*).
- **`On Long Click`**: The action/s to perform when the control is pressed for a longer time than a tap (*see screenshot above center*).
- *Additional actions:* The action/s to perform after the `On Click` or `On Long Click` actions have been executed (*see screenshot above right*). If no action has been set for the `On Click` or `On Long Click` events, then the additional action/s are performed directly on a click or long-click.

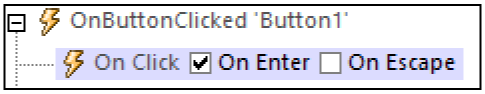
You can combine actions 667 for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.


OnButtonClicked 'Button2'
On Click ☑ On Enter □ On Escape
Update Display
Scroll To ○ Top ● Bottom ○ Row Group ○ Control
● of Page ○ of Table
On Long Click
Update Display
● Open URL ○ Open File in external App
Open URL "http://www.altova.com"
MessageBox OK
Message "Transaction completed successfully."
Title "Transaction completed."
OK
Close Subpage and optionally return result

The screenshot above shows that the `On Click` and `On Long Click` events each has a sequence of actions defined for it. An additional `MessageBox` event is defined after the `On Long Click` event. This `MessageBox` event will be executed after the `On Click` or `On Long Click` sequence of actions has completed.

*On Enter/Escape*

If the control's *On Enter* or *On Escape* check box is selected, then the control's actions are executed when the respective key (**Enter** or **Escape**) is tapped. The key-tap (**Enter** or **Escape**) serves as an alternative to the `On Click` event, and will work additionally to the click. The screenshot below shows the *On Enter* and *On Escape* check boxes of the Button event. Other controls that provide this option look similar and work similarly.


OnButtonClicked 'Button1'
On Click ☑ On Enter □ On Escape

This setting can also be accessed via the control's `On Enter/Escape` property, which is described below.

**Note:** If you select the **Page | Show/Define Tab Order**[1633] menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

## Label properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Text

The `Text` property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the Rich Text 584 control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text 1227 section for more information.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

▼ Multiline

Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

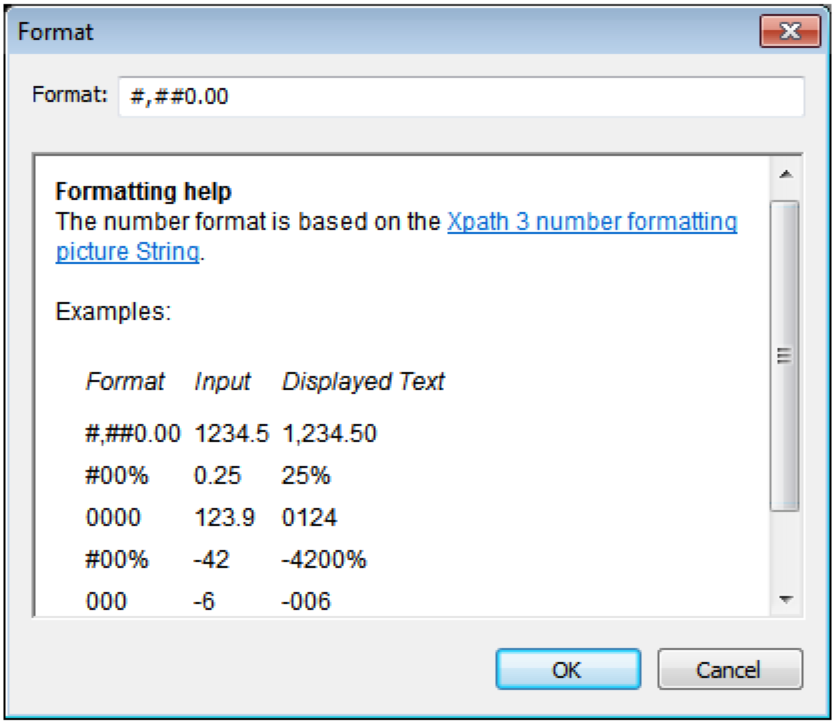
- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

▼ Max Number of Lines

This property become available only when the `Multiline` property has been set to `true`. It sets the maximum number of lines that are allowed in the control. The property's default value is `unlimited`. This property cannot be set if the `Text Size Auto-Fit` property has been enabled. You must choose either to set a maximum number of lines or to auto-fit the text.

▼ Number Format String

Click the **Additional Dialog** button and enter a number format in the Format dialog that appears (*screenshot below*).



Format: #,##0.00

**Formatting help**
The number format is based on the Xpath 3 number formatting picture String.

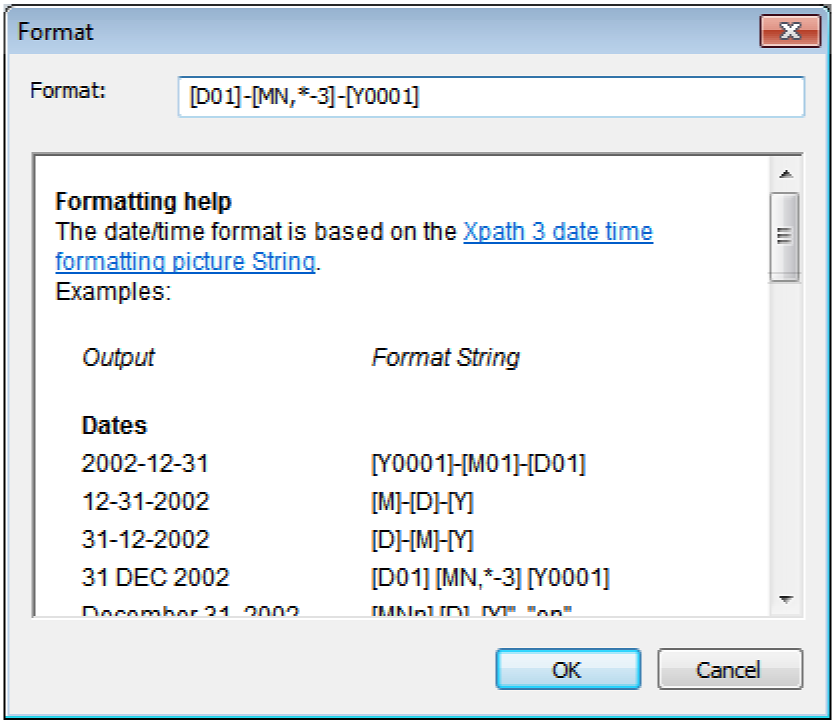
Examples:

| *Format* | *Input* | *Displayed Text* |
|---|---|---|
| #,##0.00 | 1234.5 | 1,234.50 |
| #00% | 0.25 | 25% |
| 0000 | 123.9 | 0124 |
| #00% | -42 | -4200% |
| 000 | -6 | -006 |

The formatting will be applied to the control's content if the content is numeric, and will be displayed in the solution, not in the design.

▼ Date/Time Format String

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (*screenshot below*).



[D01]-[MN,*-3]-[Y0001]

**Formatting help**
The date/time format is based on the Xpath 3 date time formatting picture String.
Examples:

| *Output* | *Format String* |
|---|---|
| **Dates** | |
| 2002-12-31 | [Y0001]-[M01]-[D01] |
| 12-31-2002 | [M]-[D]-[Y] |
| 31-12-2002 | [D]-[M]-[Y] |
| 31 DEC 2002 | [D01] [MN,*-3] [Y0001] |
| December 31, 2002 | [MNn] [D], [Y]", "en" |

The formatting will be applied to the control's content if the content has the correct lexical form of `xs:date` (for the Date control), `xs:time` (for the Time control), or `xs:dateTime` (for the Date, Time, and DateTime controls). Basic examples are:

- *xs:date:* `2014-12-31`
- *xs:time:* `23:59:59`
- *xs:dateTime:* `2014-12-31T23:59:59`

### Date/Time Format Language

Select one of the supported languages from the dropdown list of the combo box (`EN`, `DE`, `ES`, `FR`, `JA`). The selected language will be used in the date/time formatting that is set in the `Date/Time Format String` property (*see description above*). If the names of months and weekdays are used in the format string, then these will be displayed in the language selected for this property. The default language is English.

### Control Action

Click the **Additional Dialog** button to display the control's Actions dialog[667]. You can set actions to perform when a control event[665] is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog[667]. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

### ▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

### ▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

### ▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

### ▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the

**`Text Color`** property.

### Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` 1262 function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.

- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template[1200] to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Strikethrough Text

Select `true` or `false` from the dropdown list of the combo box to determine whether a line is drawn through the text of the control. The value can also be generated dynamically via an XPath expression. The default is `false`.

▼ Automatic Link Detection

A setting to specify whether a text fragment that has the pattern of a website-page URL or an email URL will be automatically displayed as a link. If the label text contains URLs (such as `www.altova.com`) or email addresses (such as `altova.user@altova.com`), then such text can be set to be automatically displayed as a live link in the label text. Clicking or tapping the link opens, respectively, the targeted website page in a browser or a new email in the device's email app.

This property can take one of the following values:

- *None:* The default value. Automatic link detection is not carried out.
- *Explicit Links:* A text fragment that begins with `http://`, `https://`, `rtsp://`, or `mailto:` is detected to be a link and is displayed as a live link in the control's text.
- *All Links:* Any text that indicates a website-page URL or email URL is automatically detected to

be a link. For example, in addition to the explicit links listed above, the following patterns would also be detected as links: (i) **`www.altova.com`**, (ii) **`altova.com`**, (iii) `altova.user@altova.com`.

### Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

### Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically

maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

### Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as high as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as high as the control's content requires.

In effect, `fill_parent` creates a maximum height, while `wrap_content` creates a minimum height.

*Default values*

- For the the Geolocation Map control, the default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.
- For all other controls, the default is `wrap_content`.

### Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**1304 and **`$MT_CanvasY`**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

On Enter/Escape

Takes one of three values:

- `OnEnter`: Specifies that the actions of this control are executed when the **Enter** key is tapped.
- `OnEscape`: Specifies that the actions of this control are executed when the **Escape** key is tapped.
- None: No action when either **Enter** or **Escape** is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to **`"OnEnter"`** or **`"OnEscape"`**. If more than one control on a page is given the same value (`OnEnter` or `OnEscape`), then the first visible and enabled control that has the value is selected when the key is tapped. (See the **`Visible`** and **`Enabled/Editable`** properties.)

This setting can also be made via the dialog to set the control's **`OnClicked`** actions (see the description of the control's events above).

**Note:** If you select the **Page | Show/Define Tab Order**1633 menu command, then controls that have been assigned an **Enter** or **Escape** key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu404.

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control1318. The dropdown list of the `Style`

Sheet property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.15 Placeholder Control

The Placeholder Control enables you to add a control template 1200 to a page as well as to a control template itself. See the section Control Templates 1200 for information about how to work with control templates and Placeholder Controls.

⊟ Notes

- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

### Placeholder Control events

There is no event associated with the Placeholder Control.

### Placeholder Control properties

The control's properties are available in the Styles & Properties Pane 274, and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

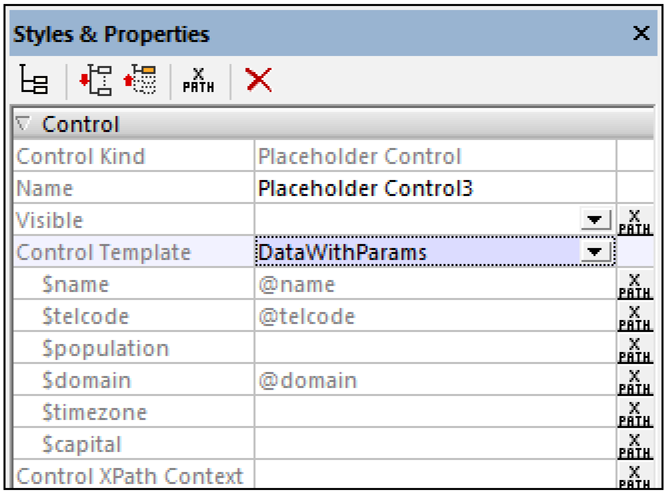
An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Control Template

Selects the control template that this Placeholder Control instantiates. The dropdown list of the property's value field displays all the control templates that have been defined in the design till this time. Alternatively, you can use an XPath expression to select a control template. Such an XPath expression must evaluate to the name of a control template in the design. Using XPath expressions enables you to select templates conditionally. See Example Projects 1210 for ways to do this.



Styles & Properties

| Control | | |
|---|---|---|
| Control Kind | Placeholder Control | |
| Name | Placeholder Control3 | |
| Visible | | XPATH |
| Control Template | DataWithParams | |
| $name | @name | XPATH |
| $telcode | @telcode | XPATH |
| $population | | XPATH |
| $domain | @domain | XPATH |
| $timezone | | XPATH |
| $capital | | XPATH |
| Control XPath Context | | XPATH |

When you select a control template, its parameters are displayed as sub-properties of the `Control Template` property *(see screenshot below)*. Enter XPath expressions to select or generate the values of the parameters. The context node of the expressions will be the context node of the Placeholder Control when the control is processed. The context node for the evaluation of parameters and variables is not changed by the `Control XPath Context` property *(see below)*.

If you select a control template via an XPath expression, then the control template is not selected till runtime, when the expression is evaluated. Since the control template is not known at design time, no parameters can be displayed in the pane. Instead of a list of parameters, a property named Template Parameters is available. You can enter an XPath expression to generate the values of the expected parameters. The expression must be either an array expression or map expression.  If you use an array expression, the parameter values must be provided in the same sequence as the parameter definition order in the control template; additionally any optional parameters must not be omitted. See Example Projects[1221] for a sample of such expressions.

## Template Parameters

The `Template Parameters` property provides the parameter values for the control template selected in the Placeholder control's `Control Template` property.

For the `Template Parameters` property, you can enter an XPath expression to generate the values of the expected parameters. The XPath expression must be either an array expression or a map expression. If you use an array expression, the parameter values must be provided in the same sequence as the parameter definition order in the control template; additionally any optional parameters must not be omitted. (In the case of maps, the keys of the map enable the values to be correctly assigned.) See Control Templates Example Projects[1221] for a sample of such expressions.

## Control XPath Context

This property is used to change the context node of XPath expressions that are evaluated in the control template that the Placeholder Control instantiates. The instantiated control template is the template named in the `Control Template` property *(see above)*.

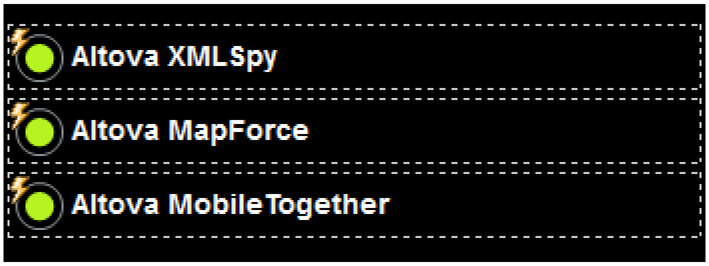
Note the following points:

- XPath expressions for the values of template parameters are evaluated in the context of the Placeholder Control's context node—not the changed context node.
- XPath expressions that generate the values of template variables, however, will be evaluated in the context specified by this property (`Control XPath Context`). If no value is set for this property, then values of variables are also evaluated using the Placeholder Control's context node.

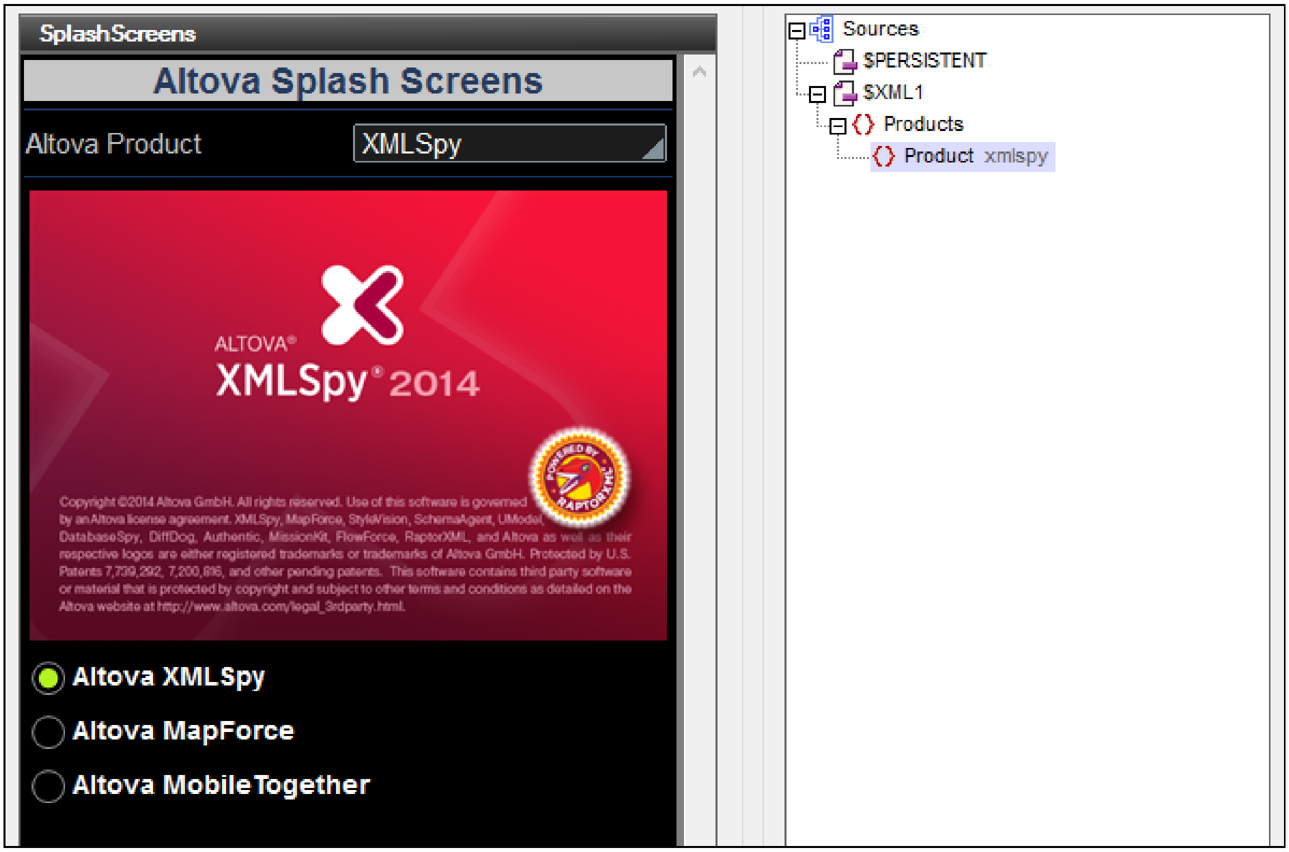
## 9.1.16 Radio Button

Radio buttons can be used to constrain the user to select one of a set of specific values. Each radio button is associated with one specific value. A set of radio buttons are grouped into a mutually exclusive set by associating all of them with a single page source node. By "mutually exclusive" is meant that the user is constrained to select only one radio button from the set. The value of the radio button that is selected will be entered into the associated page source node. When you insert a radio button control, you can specify whether the radio button should be to the left or right of the button text, or in the default system position.

For example, in the screenshot below, all three radio buttons are associated with one page source node: an element called `Product`. This creates a set of mutually exclusive radio buttons for the `Product` node. Each radio button is given a value through its `Checked Values` property. The button text is the value of the button's `Text` property.



When the solution is run, all three radio buttons are initially displayed empty (*see screenshot below*). When the user selects a radio button, that button's checked value (the product name in our example) is passed to the associated node (the `Product` element; *see the XML Data tree in the screenshot below*).



Copyright ©2014 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,739,292, 7,200,816, and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at http://www.altova.com/legal_3rdparty.html.

Two key properties of the radio button are:

- The text that accompanies the radio button. This can be static text (entered as the value of the `Text` property; *see below*) or a dynamic value obtained via an XPath expression.
- The checked value of the radio button is assigned with the `Checked Values` property (*see below*).

Radio buttons have the `OnFinishEditing` event, which is triggered when the end-user selects the radio button. To define an action for this event, click the **Additional Dialog** button of the `Control Action` property. This displays the Control Actions dialog 667, in which you can specify the required action. In our example, the event triggers the a reload of the splashscreen image.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View 253) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.

- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

## Radio Button events

The **`OnFinishEditing event`**[665] is available. For a description of the actions that can be defined for this event, see the Actions section[667].

## Radio button properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Text

The `Text` property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **`XPath`** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the Rich Text[584] control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text[1227] section for more information.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

### Multiline

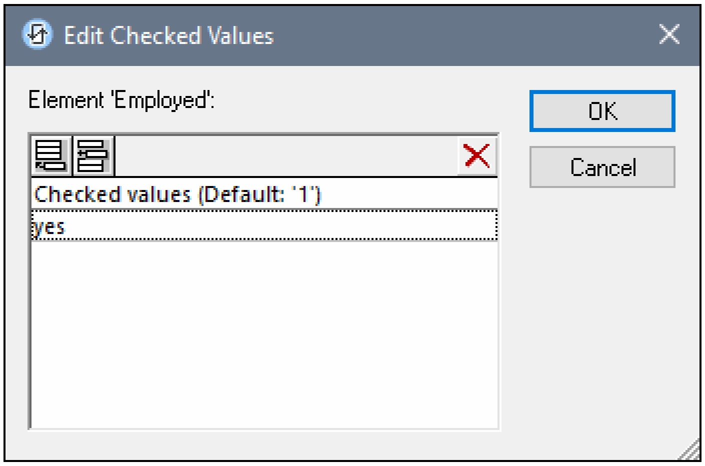
Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

### Checked Values

Provides an XML data value for the selected state of the control. The default value is **`1`**. To change the XML data value, click the property's **Additional Dialog** button. In the Edit Checked Values dialog that appears (*screenshot below*), enter a value for the checked (selected) state. If the radio button is selected by the end-user, the XML data value will be entered as data in the XML node associated with the control.



Element 'Employed':

Checked values (Default: '1')
yes

Note the following points:

- To specify which node will receive the value, make a page source link from the control to a page source node (by dragging the node on to the control).
- If you enter more than one value for the checked value, then the first value will be used; the others are ignored.
- You can enter a value for the *Checked Values* property in a style sheet 1318 for all radio buttons.
- If the *Checked Values* property is defined in a style sheet 1318, and a control is, as a consequence, assigned values for this property in more than one place, then the most local definition wins. The most local definition is the style defined directly in the control's property. For priority involving a style sheet, see Priority within a Style Sheet 1321 and Priority across Style Sheets 1325.

Get Value from XPath

The value returned by the XPath expression is displayed in the control. This enables alternative ways to enter display values for certain controls. For example, a combo box could take its display value from either a page source node or the return value of the **`Get Value from XPath`** property.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the generation of the value of the `Get Value from XPath` property. If used, then a validation error results.

Auto Correct Value

The control has two states: checked and unchecked, each of which is associated with an XML value. These XML values are defined in the `Checked Values` property. The control is also associated with a page source node. If the value in the page source node is anything other than the XML value defined for the current status of the control (checked or unchecked), then auto-correction updates the value in the node to the value that has been defined for the current status.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node 886 action. The `Auto Correct Value` property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

For example, if the checked value is defined to be **`1`** and the unchecked value to be **`0`**, and if the node's content is **`"somestring"`**, then, (i) if the control is checked, the node's content will be corrected to **`1`**, (ii) if the control is unchecked, the node's content will be corrected to `0`. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction.

The `Auto Correct Value` property has two possible values: `true` or `false`. If the property is set to `true`, XML values are automatically corrected. The property's default value is `false`.

▼ Check Mark Position

Sets the position of the check box relative to the control's text: either to the left or right of the text. The default is the operating system default.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

▼ Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` 1262 function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**1304 and **`$MT_CanvasY`**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template1200 to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

▼ Strikethrough Text

Select `true` or `false` from the dropdown list of the combo box to determine whether a line is drawn through the text of the control. The value can also be generated dynamically via an XPath expression. The default is `false`.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as high as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as high as the control's content requires.

In effect, `fill_parent` creates a maximum height, while `wrap_content` creates a minimum height.

*Default values*

- For the the Geolocation Map control, the default is the smaller of the two values **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.
- For all other controls, the default is `wrap_content`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**1304 and **`$MT_CanvasY`**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order**1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

### Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu404.

### Style Sheet

The `Style Sheet` property sets the style sheet to use for the control1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see [Applying User-Created Style Sheets](1327)*[1327]). See the section Style Sheets[1318] for more information.

Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.17 Rich Text

The Rich Text control enables you to display formatted text from page sources in which the text is marked up with formatting (styles). Furthermore, on web clients and Windows PC clients, text in a Rich Text control can be edited and formatted, and saved back to the page source. For a broad description of the Rich Text feature, see the section Design Object/Features | Rich Text[1225].

Notes

- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

### Rich Text events

The **`OnFinishEditing event`**[665] is available. For a description of the actions that can be defined for this event, see the Actions section[667].

### Rich Text properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### ▾ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### ▾ Text

The `Text` property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the Rich Text[584] control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text[1227] section for more information.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Hint

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Text Color Hint

Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (*see above*). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Rich Text Style Sheet

Specifies which of the design's Rich Text style sheets is to be used for the layout of text in the Rich Text control. The dropdown list of the property's combo box displays all the project's Rich Text style sheets (created via the Rich Text Style Sheets 1598 dialog).

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

### Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

## Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

## Rich Text Height

Specifies the height of the Rich Text control. The default behavior (*wrap_content*) is for the height of the control to increase as content increases, up to a maximum of the screen height on devices and browser window on web clients. When the maximum height is reached, the content part of the control displays a scrollbar; the toolbar part is fixed at the top of the control. Alternatively, you can set a fixed height (in pixels, dp, or sp) for the control.

## Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**1304 and **`$MT_CanvasY`**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order**1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

### Style Sheet

The `Style Sheet` property sets the style sheet to use for the control1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*1327). See the section Style Sheets1318 for more information.

### Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property384.

## 9.1.18 Signature Field

The Signature Field control enables the signature of an end user to be stored as a graphic file. This is useful, for example, in courier-business solutions, where signatures are used to acknowledge receipt of a couriered item. As the end user starts drawing his signature in the signature field, the signature is written in Base64 image encoding to a page source node. When the page source is saved, the Base64 image data is saved in it, in the node designated for it.

The signature image has the following default properties. Its background-color is the inverse of the page background-color. The signature itself is the same color as the page background-color. The image width is the smaller of the device's viewport dimensions. The image height is half of its width. How these values are calculated with XPath expressions is shown in the table below, together with the control properties you can use to customize these settings.

| Signature property | Default value | Custom value via control property... |
|---|---|---|
| *Signature color* | `$MT_PageBackgroundColor`[1300] | `Text Color` |
| *Signature background-color* | `mt-invert-color`[1262] (`$MT_PageBackgroundColor`[1300]) | `Background Color` |
| *Signature image width* | `min ($MT_CanvasX`[1304]`, $MT_CanvasY`[1304]`)` | `Signature Creation Width` |
| *Signature image height* | `min ($MT_CanvasX`[1304]`, $MT_CanvasY`[1304]`) div 2` | `Signature Creation Height` |

The main settings for the signature field are:

- a page source link, which is the page source node where the signature image data is stored. Drag a page source node onto the control to create/modify the control's page source link. Delete the page source link to clear the association (see *Notes* below).
- the `Signature Creation Width` and `Signature Creation Height` properties; these specify the dimensions of the image that will be created
- the `Text Color` and `Background Color` properties; these specify the colors of the signature text and its background
- some *Save* action that saves the signature image data to the page source; till such an action is executed, the data is stored only in the temporary XML tree

Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the

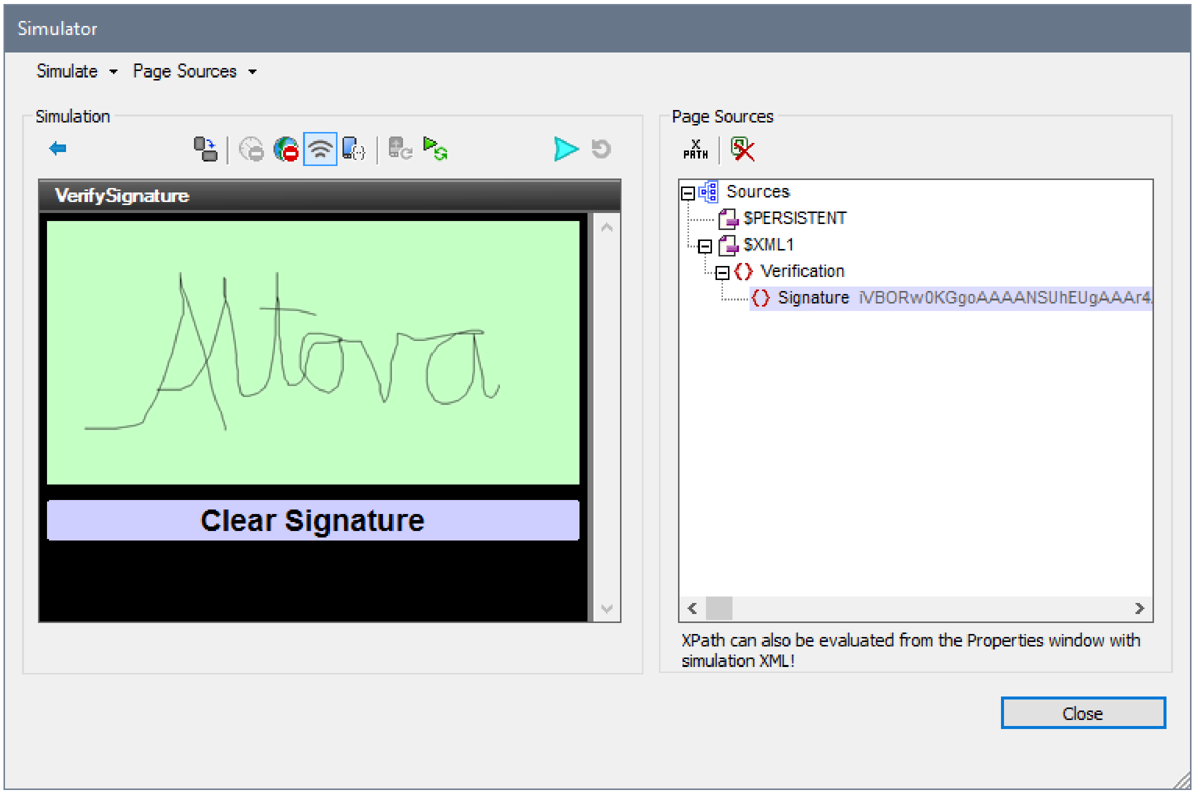
associated control.

- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View 253) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog 296).
- A control's CSS properties can be defined in the Styles & Properties Pane 274 and/or in an external CSS file 296. Those defined in the Styles & Properties Pane 274 have priority.

## Enabling the end user to edit a signature

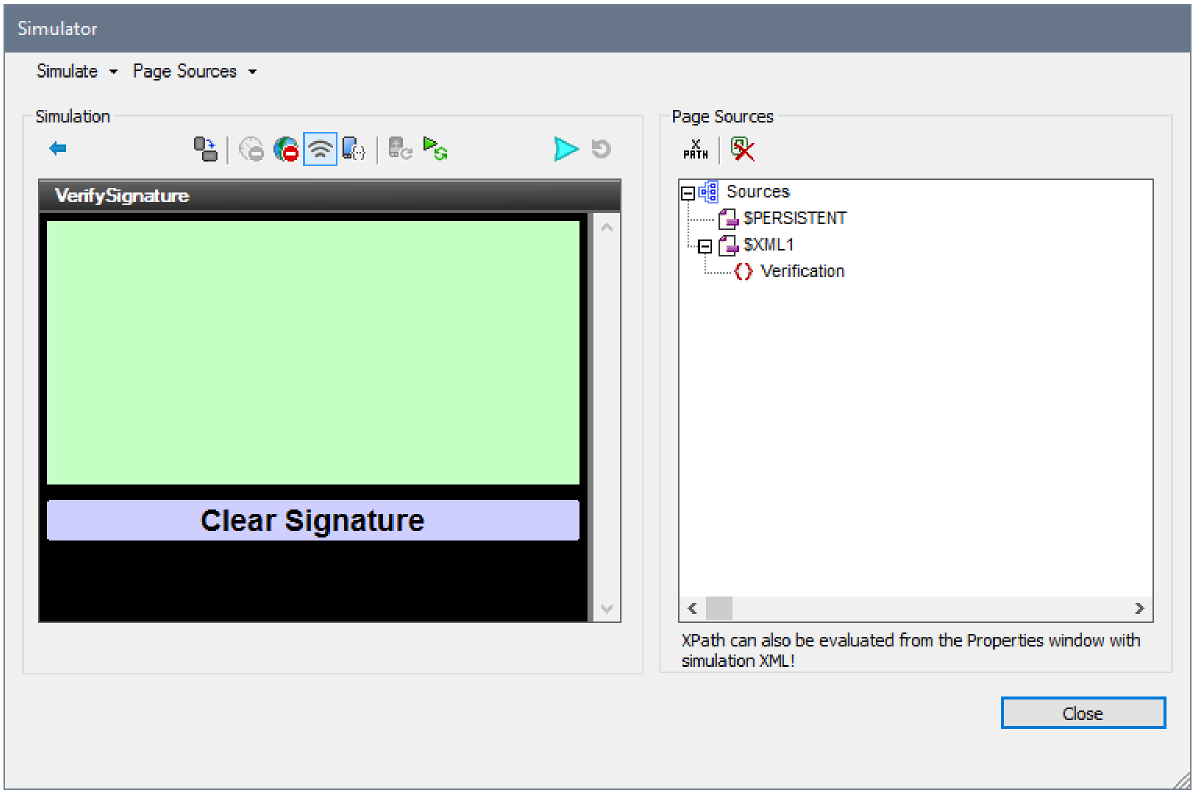
The end user's signature is created as an image in a page source node. This being the case, an end user can only add data to a signature drawing that has been started. If the image needs to be edited—for example, if the end user has drawn the signature unsatisfactorily—then the image data must be removed from the node (or the node itself must be deleted), and the signature must be re-drawn. The simplest way to implement this functionality is to create a button control that deletes the node. Do this as follows:

1. Create a button control near the signature field control (*see screenshot below*).
2. Add a Delete Node 879 action as the button's `OnClick` event, and set the signature's page source node as the node to delete.



XPath can also be evaluated from the Properties window with simulation XML!

3. Test the button in a simulation. In the screenshot above, notice that the signature is drawn and the image data is saved to the `Signature` node. The screenshot below was taken after the button was clicked. Notice that the node has been deleted and the signature field has therefore been cleared.



XPath can also be evaluated from the Properties window with simulation XML!

4. If a signature is now drawn in the signature field, then the `Signature` node is re-created with the image data of the new signature.

**Note:** Alternatively, you can set the button action to update the signature's page source node with the empty string (*see the Update Node*[886] *action*). This would delete the image data from the node and therefore clear the signature field, but the node itself would not be deleted.

## Signature Field events

There is no event associated with the Signature Field control.

## Signature Field properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

### Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▾ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

▾ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▾ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▾ Image Anti-Aliasing

Defines whether anti-aliasing is used when the image associated with the control is created. (Anti-aliasing is a technique for removing the jagged edges in images.) The property can take a value of `true` or `false`. The default is `false`. *Tip*: If image colors are going to be converted subsequently, it is advisable to set the property's value to `false`. This is because anti-aliasing color information in the original image cannot reliably be converted to suitable anti-aliasing colors in the target color scheme. *Note:* For web clients, the value of this property is ignored, and anti-aliasing is always in effect.

▼ Signature Creation Width

A number, in pixels, dp, or sp, that specifies the width of the signature image. It is the width of the canvas on which the signature is drawn. For example, a value of `400` will assign a width of 400 pixels to the signature image. Note that this is the width of the image that is created, and is not necessarily that of the image rendered on the mobile device. The rendered image is scaled to the width defined in the **`Control Width`** property. For example, if the signature image has a width of 400 pixels, and the **`Control Width`** property has a value of `80%`, then, on a device that has a width of 1000 pixels, the image will scale up to a width of 800 pixels (80% of the device width).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Signature Creation Height

A number, in pixels, dp, or sp, that specifies the height of the signature image. It is the height of the canvas on which the signature is drawn. For example, a value of `200` will assign a height of 200 pixels to the signature image. Note that this is the height of the image that is created; it is not necessarily the height of the image rendered on the mobile device. The height of the rendered image is scaled proportionally to the width defined in the **`Control Width`** property. For example, if the signature image has a height of 300 pixels and a width of 400 pixels, and the **`Control Width`** property has a value of `80%`, then, on a device that has a width of 1000 pixels, the image will scale up to a width of 800 pixels (80% of the device width) and to a height of 600 pixels (which maintains the image's aspect ratio).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

### Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as high as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as high as the control's content requires.

In effect, `fill_parent` creates a maximum height, while `wrap_content` creates a minimum height.

*Default values*

- For the the Geolocation Map control, the default is the smaller of the two values **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.
- For all other controls, the default is `wrap_content`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

## 9.1.19 Space

The Space control enables you to add vertical space to the design. This is useful if you wish to have precise control over the space between design components.

Notes

- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

### Space events

There is no event associated with the Space control.

### Space properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

## Space Height

Sets the height in pixels, dp, or sp of the selected space object. Select a height value (in pixels, dp, or sp) from the dropdown list of the combo box, or double-click in the value field to enter a numeric value, or use an XPath expression. The numeric value is understood to be a pixel value. For this reason, no unit should be specified.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

### ⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

## Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▾ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▾ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▾ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▾ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.20 Switch

A switch is associated with a page source node. The end user can choose between two states: *selected* or *unselected*. Depending on which state is chosen, a corresponding XML value is entered as content of the associated node. The values corresponding to the respective states (*selected* or *unselected*) are defined in the `Checked Values (true/false)` property (*see property description below*). A typical use case would be: first, let the end-user's switch selection determine the content of a node; second, let the content of this node determine a page action, such as, whether a control is enabled/editable.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

### Switch events

The **`OnFinishEditing event`**[665] is available. For a description of the actions that can be defined for this event, see the Actions section[667].

### Switch properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Multiline

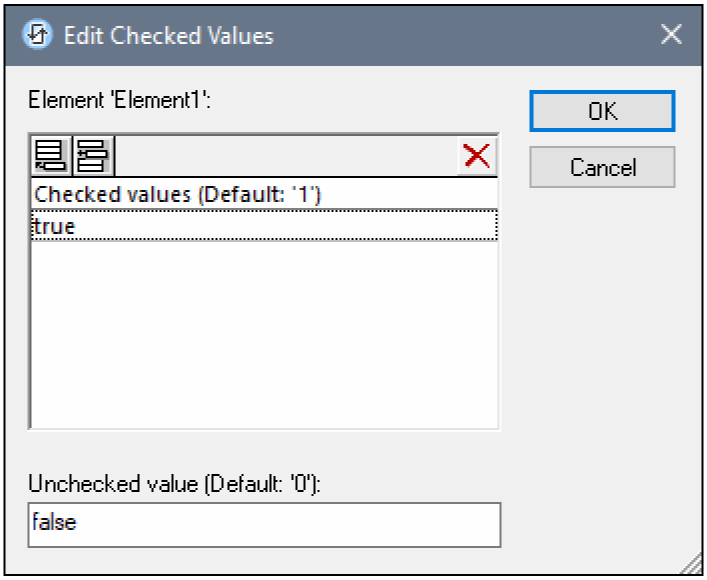
Sets multiline input/display on or off (`true/false`). The default is `false`. If the text of the control is longer than a single line, and the value is **`true`**, then the text will wrap to a new line; if the value is **`false`**, then the text will be truncated at the end of the first line.

*Note*

- If this property is set to **`true`**, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to **`true`** on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

### Checked/Unchecked Values

Provides an XML data value for the selected/unselected state of the control. The defaults are, respectively, `1` and `0`. To change the XML values that will be entered in the associated page source node, click the property's **Additional Dialog** button. In the Edit Checked Values dialog that appears (*screenshot below*), enter a value for each the checked (selected) and unchecked (unselected) state. When the end user changes the control's state, the value corresponding to the new state (selected/unselected) will be entered in the page source node.



Element 'Element1':

Checked values (Default: '1')

true

Unchecked value (Default: '0'):

false

Note the following points:

- To specify which node will receive the value, make a page source link from the control to a page source node (by dragging the node on to the control).
- If you enter more than one value for the *Checked* value, then the first value will be used; the others are ignored.
- If you enter a *Checked* value but do not enter an *Unchecked* value, then the *Unchecked* value is no longer the default value but an empty string.
- You can enter values for the *Checked/Unchecked Values (true/false)* property in a style sheet[1318] for: (i) all check boxes, and/or (ii) all switches, and/or (iii) all controls that use *Checked/Unchecked Values (true/false)* (which are check boxes and switches).
- If values for *Checked/Unchecked Values (true/false)* are defined in a style sheet[1318], and a control is, as a consequence, assigned values for this property in more than one place, then the most local definition wins. The most local definition is the style defined directly in the control's property. For priority involving a style sheet, see Priority within a Style Sheet[1321] and Priority across Style Sheets[1325].

### Get Value from XPath

The value returned by the XPath expression is displayed in the control. This enables alternative ways to enter display values for certain controls. For example, a combo box could take its display value from either a page source node or the return value of the **`Get Value from XPath`** property.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the generation of the value of the `Get Value from XPath` property. If used, then a validation error results.

### Auto Correct Value

The control has two states: checked and unchecked, each of which is associated with an XML value. These XML values are defined in the `Checked Values` property. The control is also associated with a page source node. If the value in the page source node is anything other than the XML value defined for the current status of the control (checked or unchecked), then auto-correction updates the value in the node to the value that has been defined for the current status.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node 886 action. The `Auto Correct Value` property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

For example, if the checked value is defined to be **`1`** and the unchecked value to be **`0`**, and if the node's content is **`"somestring"`**, then, (i) if the control is checked, the node's content will be corrected to **`1`**, (ii) if the control is unchecked, the node's content will be corrected to **`0`**. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction.

The `Auto Correct Value` property has two possible values: `true` or `false`. If the property is set to `true`, XML values are automatically corrected. The property's default value is `false`.

▼ Toggle On Text

Sets the text that appears in the control when the control is toggled on. Double-click inside the value field to edit.

▼ Toggle Off Text

Sets the text that appears in the control when the control is toggled off. Double-click inside the value field to edit.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

▼ Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

▼ Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height`[1262] function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template 1200 to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the Multiline property has been set to **true**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's Wrap Content Auto-Fit Group property.

▼ Bold Text

Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

▼ Italic Text

Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

▼ Underline Text

Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, #FF0000), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the **Background Color (Disabled)** property.

▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the **XPath**

toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** 1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

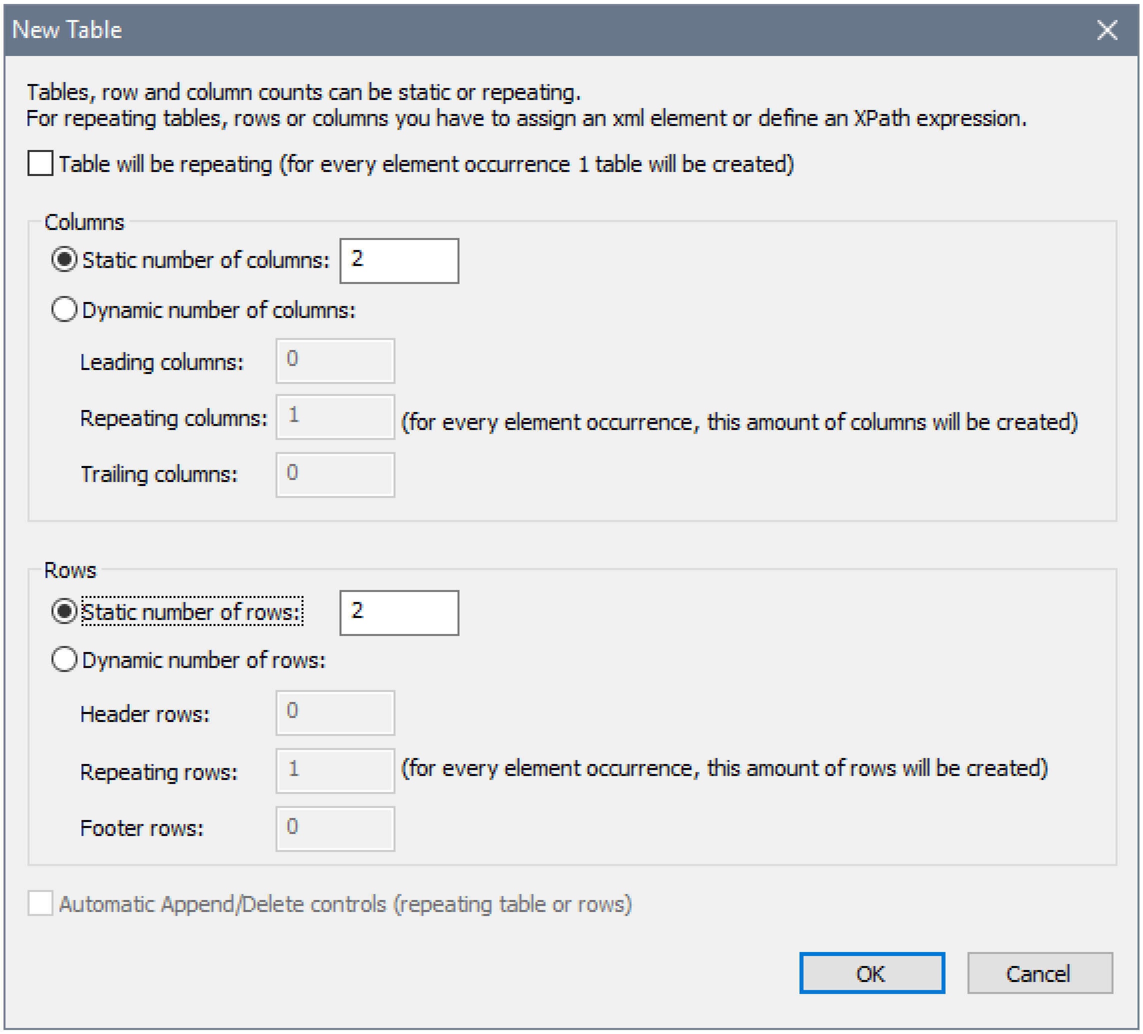
You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.21 Table

The Table control inserts a static table[1063], repeating table[1065], a table with dynamic rows[1069], or a table with dynamic columns[1074]. On dropping the control in the design, the New Table dialog (*screenshots below*) appears.

- If you specify a static number of columns and rows (*screenshot below left*), a static table[1063] is created.
- A repeating table[1065] is one in which a new table is created for every occurrence of the element that is associated with the table. To create a repeating table[1065], select the *Table will be repeating* check box. Each instance of the element that repeats is created as a new table. The number of columns and rows are specified as static numbers (*see screenshot below right*).
- In a table with dynamic rows[1069], it is not the entire table, but a table row group, that repeats. To create this kind of table, ensure that the *Table will be repeating* check box is unselected, and then select the *Dynamic number of rows* radio button. Each instance of the element that repeats is created as a user-specified number of rows. The number of columns can be fixed (static) or repeating (dynamic).
- In a table with dynamic columns[1074], within each row, a column corresponding to the selected element node repeats. In this way, a new column is created dynamically for each occurrence of the corresponding element. To create dynamic columns, select the *Dynamic number of columns* radio button. Dynamic columns can be generated for repeating tables[1065] as well as for tables with dynamic rows[1069].



New Table

Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

□ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 2

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 2

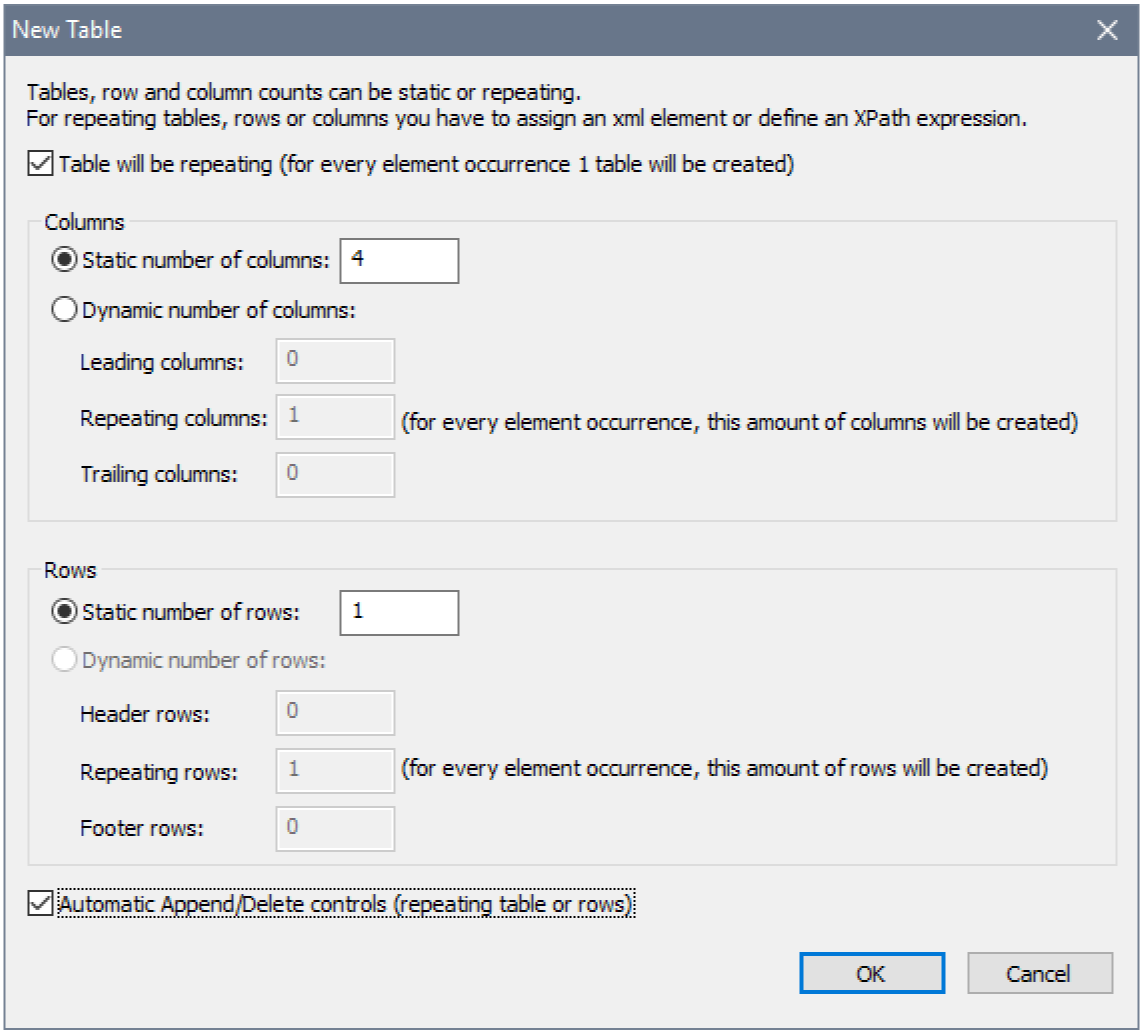
○ Dynamic number of rows:

Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

□ Automatic Append/Delete controls (repeating table or rows)



Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

☑ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 4

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 1

○ Dynamic number of rows:

Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

☑ Automatic Append/Delete controls (repeating table or rows)

Repeating tables and tables with dynamic rows can also have Append/Delete controls automatically added. If added, each instance of a repeating element will have a Delete control next to it. This allows the end user to delete this instance of the element. Depending on whether each instance of the repeating element is created as a table (in repeating tables) or a row (in tables with dynamic rows), the Add control enables a new table or row (and hence an instance of the corresponding element) to be added.

The cells of the table (both static and dynamic) can contain the following:

- Static text
- A node from a page source
- A page control
- A nested table

Table formatting properties are available in the Styles & Properties Pane[274] and in the context menu of the table.

For more information, see the section Tables[1062].

Notes

- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

## Table events

There is no event associated with the Table control.

## Table properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

*Table Cell*

All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Skip wrap_content

This property is available for table cells, and takes a value of `true` or `false`, with the default being `false`. If set to `true`, this cell's content will be ignored when making the `wrap_content` calculation to determine the containing column's minimum width.

▼ Min Cell Height

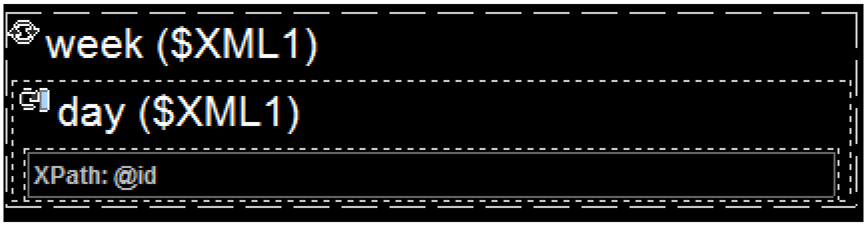
The `Min Cell Height` property sets the minimum height of the cell. Select a value from the property's combo box or enter a value directly. The height may be given in any of the following units: *pixel, dp,* or *percentage of original cell height* . The value must include the length unit, similarly to the entries in the combo box. If the cell content has a height less than that specified by `Min Cell Height`, then the cell height will be that given by this property. If the cell content has a height more than that specified by `Min Cell Height`, then the cell height will be that of the cell content.

▼ Spans Column Groups

This property is available only in the first column of a column group[1074]. It takes a value of **`true`** or **`false`**. Default is **`false`**. If the property is set to **`true`**, then all the columns of all column groups are spanned to a single column, whether the column group in the design consists of one or more columns.

You can think of the transformation process as occurring in two steps: (i) In the design, all columns (of any type, including static) in the column group are joined together into one column, as if the **Join**[1087] command[1087] were used on them; (ii) in the output, all the instances of the repeating element are created as a single column. Any XPath expression that: (i) is located within a column in the design, and (ii) tries to locate individual element instances corresponding to output columns will return an error.

The screenshot below shows a simple example of a dynamic column created in a column group. The column group in the design contains a single column group that is associated with the **`day`** element, and this column group is within a (repeating) table[1065] associated with the **`week`** element (which, in the page source, is the parent of the **day** element). Since the **week** element repeats, a new table will be created for each **week** element. If, in the page source, there are multiple **day** child elements of the **week** element, and if, in the design, the dynamic columns of the column group are not spanned, then the table (for each **week**) generated from this design will have as many columns as there are **day** child elements. If, however, you set the `Spans Column Groups` 614 property to `true`, then the columns in the generated table will be spanned, and the table will have only one column.



For more information about column groups, see Dynamic Columns 1074. For details of how to span columns, see the section Row/Column joining and spanning 1083.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Border Width

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the `Border Width` property to display the individual border width properties (left, right, top and bottom), and set the different value. For example: If you set `Border Width` to be `1px` and `Border Width Top` to be `2px`, then the widths of the left, right and bottom borders will all be `1px`, while the top border width will be `2px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

#### *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Border Color

Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the `Border Color` property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set `Border Color` to be `blue` and `Border Color Top` to be `red`, then the color of the left, right and bottom borders will all be `blue`, while the top border color will be `red`.

### Border Style

Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four sides, expand the `Border Style` property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set `Border Style` to be `dashed` and `Border Style Top` to be `solid`, then the style of the left, right and bottom borders will all be `dashed`, while the top border style will be `solid`.

### ▾ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

*Table Column*

### ▾ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog 296 has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane 274 will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane 274.
- You can also specify a style sheet to use, as shown in the second map above.

### ▾ Width

Sets the width of the object. Select `fill_parent` or `wrap_content` or a percentage value from the dropdown list of the combo box. Percentage values are percentages of the parent object.

- `fill_parent`: the width is as wide as the parent, which could be, for example, a table row
- `wrap_content`: the width is only as wide as required by the content; when this value is selected, the property `Max Width` becomes available. If you want to specify that a particular cell's width not be considered for the calculation of a table-column's width, then set the `Skip wrap_content` property of the relevant cell to `true`.
- `wrap_content_fit_parent`: If a table is broader than its parent object, then this value can be used on a column to reduce the column's width. If the table is less wide than its parent object, this value has the same effect as `wrap_content`.
- *percent value*: a percentage of the parent object; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel value, dp value, or sp value from the dropdown list, or enter a value directly

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Width

This property is available only when the table column's **`Width`** property has been set to **`wrap_content`**. The `Max Width` property sets the maximum width of table columns. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—

and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**1304 and **$MT_CanvasY**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Border Width

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the `Border Width` property to display the individual border width properties (left, right, top and bottom), and set the different value. For example: If you set `Border Width` to be `1px` and `Border Width Top` to be `2px`, then the widths of the left, right and bottom borders will all be `1px`, while the top border width will be `2px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX**1304 and **$MT_CanvasY**1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Border Color

Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the `Border Color` property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set `Border Color` to be `blue` and `Border Color Top` to be `red`, then the color of the left, right and bottom borders will all be `blue`, while the top border color will be `red`.

▼ Border Style

Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four sides, expand the `Border Style` property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set `Border Style` to be `dashed` and `Border Style Top` to be `solid`, then the style of the left, right and bottom borders will all be `dashed`, while the top border style will be `solid`.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

⊟ *Note about background colors*

If different background colors are given to a table row and a table column, the cell at the intersection gets the color of the row. This can be overridden by setting a background color on the cell.

*Table Row*

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog 296 has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }
```

```
map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane 274 will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane 274.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Border Width

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the `Border Width` property to display the individual border width properties (left, right, top and bottom), and set the different value. For example: If you set `Border Width` to be `1px` and `Border Width Top` to be `2px`, then the widths of the left, right and bottom borders will all be `1px`, while the top border width will be `2px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Border Color

Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the `Border Color` property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set `Border Color` to be `blue` and `Border Color Top` to be `red`, then the color of the left, right and bottom borders will all be `blue`, while the top border color will be `red`.

▼ Border Style

Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four sides, expand the `Border Style` property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set `Border Style` to be `dashed` and `Border Style Top` to be `solid`, then the style of the left, right and bottom borders will all be `dashed`, while the top border style will be `solid`.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

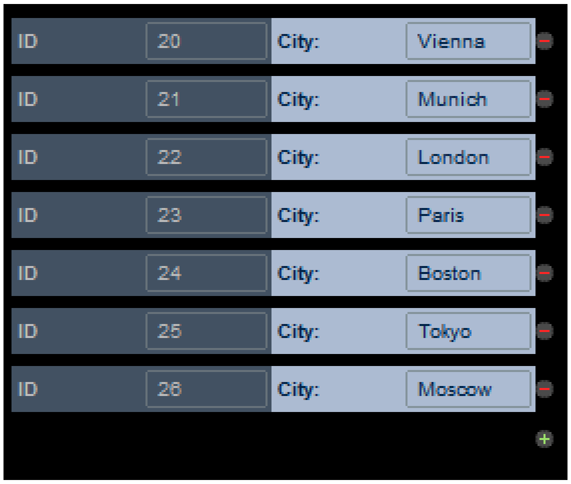
⊟ *Note about background colors*

If different background colors are given to a table row and a table column, the cell at the intersection gets the color of the row. This can be overridden by setting a background color on the cell.

*Table Row Group*

### Automatic Append/Delete Controls

Automatically adds append/delete controls to (i) tables with a repeating table structure, and (ii) rows in a table with a repeating row structure. The **Append** button *(see screenshot below)* appends another occurrence of the repeating structure. The **Delete** button is available for each occurrence of a repeating structure, and deletes that occurrence. The screenshot below shows these buttons in the repeating table structure of a solution.



| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

This property defines whether the **Append** and **Delete** buttons are automatically added to the design. The property can have values of `true` and `false`. The default is **`true`**.

### Create For Each Item In

This property is available for tables, row groups, and column groups. It sets the number of times the item (table, row, or column) repeats to the number of items returned by the property's XPath expression.

The XPath expression can select a repeating node in a data tree. This is useful if you want the table, row, or column to be generated as many times as a certain node occurs and want to set the number of these repeats dynamically. In this way, if the number of the selected node's occurrences changes, then the number of repeats is automatically modified. For example, if the property's XPath expression is `$XML1/Office/Department`, then the number of times that the table, row, or column is generated will be equal to the number of `$XML1/Office/Department` elements.

The XPath expression can also be unrelated to a data tree. In this event, the table is created once for each item in the returned sequence. For example:

- 9 times for the expression `1 to 9`
- 2 times for `"John", "Mary"`
- 2 times for `45, true()`

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—

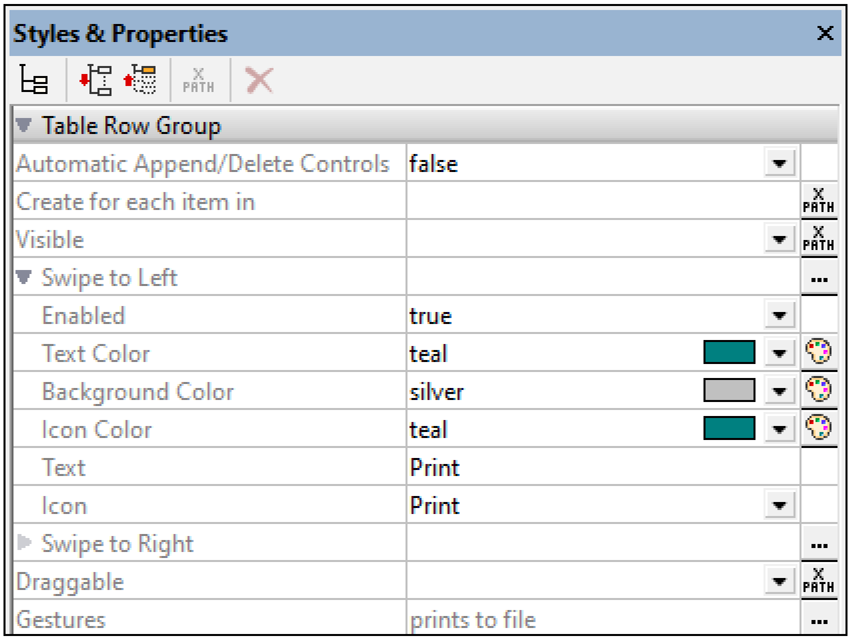
and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Swipe to Left

This property is set on a row group of a top-level table and applies to the rows of that row group. When the end user swipes left on a row of the row group, a text string and/or an icon can be displayed in the row area uncovered by the swipe. Define the corresponding text and/or icon properties in the sub-properties of this property (*see screenshot below*). The icon is selected from a set of available icons. Note that the values of properties can also be generated dynamically via XPath expressions.



Styles & Properties

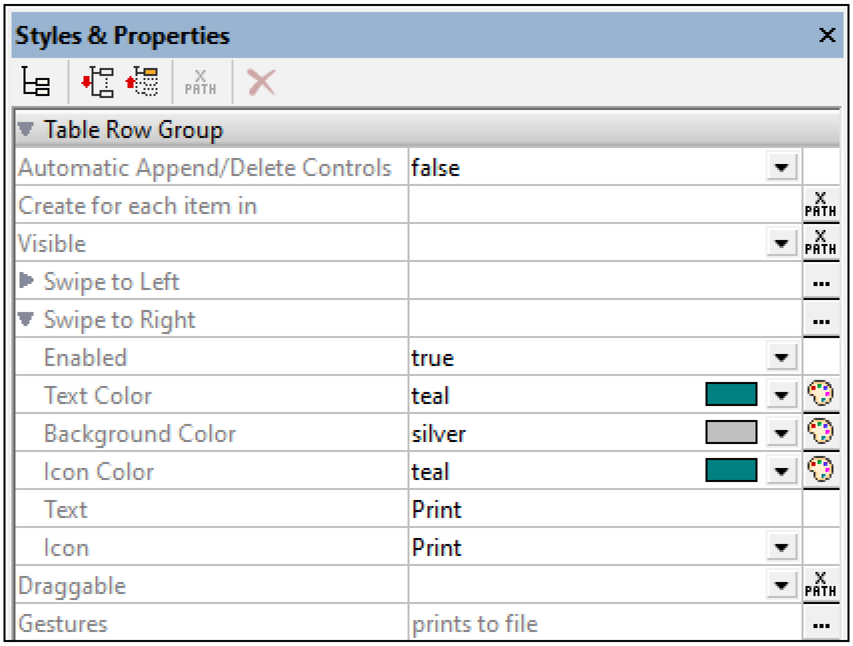
| | |
|---|---|
| ▼ Table Row Group | |
| Automatic Append/Delete Controls | false |
| Create for each item in | |
| Visible | |
| ▼ Swipe to Left | |
| Enabled | true |
| Text Color | teal |
| Background Color | silver |
| Icon Color | teal |
| Text | Print |
| Icon | Print |
| ▶ Swipe to Right | |
| Draggable | |
| Gestures | prints to file |

To set actions to execute when the end user swipes, click the **Additional Dialog** button of the `Swipe to Left` property and set actions in the Actions dialog. Alternatively, you can open the Actions dialog via the **Additional Dialog** button of the `Gestures` property.

See the topic Table Properties[1078] for more information.

▼ Swipe to Right

This property is set on a row group of a top-level table and applies to the rows of that row group. When the end user swipes right on a row of the row group, a text string and/or an icon can be displayed in the row area uncovered by the swipe. Define the corresponding text and/or icon properties in the sub-properties of this property (*see screenshot below*). The icon is selected from a set of available icons. Note that the values of properties can also be generated dynamically via XPath expressions.



Styles & Properties

Table Row Group

| | |
|---|---|
| Automatic Append/Delete Controls | false |
| Create for each item in | |
| Visible | |
| Swipe to Left | |
| Swipe to Right | |
| Enabled | true |
| Text Color | teal |
| Background Color | silver |
| Icon Color | teal |
| Text | Print |
| Icon | Print |
| Draggable | |
| Gestures | prints to file |

To set actions to execute when the end user swipes, click the **Additional Dialog** button of the `Swipe to Right` property and set actions in the Actions dialog. Alternatively, you can open the Actions dialog via the **Additional Dialog** button of the `Gestures` property.

See the topic Table Properties 1078 for more information.

## Draggable

This property determines whether the rows of a top-level row group in a table may be dragged to another location in the row group. It enables the end user to change the order of rows in a row group. Values are `true` or `false`. The default is `false`. If you set the value of the `Draggable` property to `true`, then a popup appears asking whether you want to automatically let MobileTogether define the reordering actions. Unless you wish to assign other actions *instead* of reordering actions, select **Yes**. The property's value can also be generated dynamically via an XPath expression. See the topic Table Properties 1078 for more information.

## Gestures

Click the **Additional Dialog** button to open an Actions dialog in which you can set actions to execute when the end user swipes a row of a row group left/right or drags a row to a new position within the row group. You can select one or more of the regular actions to execute. See the description of individual actions 667 for information about them. See the topic Table Properties 1078 for more information about swiping and dragging. The actions that you select will be entered as the value of the `Gestures` property.

*Table Column Group*

## Create For Each Item In

This property is available for tables, row groups, and column groups. It sets the number of times the item (table, row, or column) repeats to the number of items returned by the property's XPath expression.

The XPath expression can select a repeating node in a data tree. This is useful if you want the table, row, or column to be generated as many times as a certain node occurs and want to set the number of these repeats dynamically. In this way, if the number of the selected node's occurrences changes, then the number of repeats is automatically modified. For example, if the property's XPath expression is `$XML1/Office/Department`, then the number of times that the table, row, or column is generated will be equal to the number of `$XML1/Office/Department` elements.

The XPath expression can also be unrelated to a data tree. In this event, the table is created once for each item in the returned sequence. For example:

- 9 times for the expression `1 to 9`
- 2 times for `"John", "Mary"`
- 2 times for `45, true()`

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties[1078].

**Note:** The `$MTControlValue`[1304] variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

*Table*

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
    }

map{
    "Style Sheet"      : "Sheet-1"
    }
```

Note the following points:

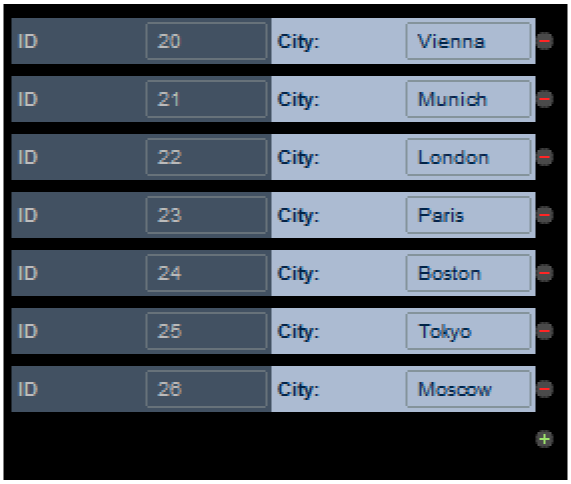
- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

## Repeating

Defines whether the table is repeating[1065] or static[1063]. Its values are `true` or `false`. The value of this property is automatically assigned at the time a repeating (=`true`) or static table (=`false`) is created. After a table has been created as a particular type (repeating or static), its type can be changed by changing the value of the **`Repeating`** property. The **`Repeating`** property is not available for dynamic tables[1069].

## Automatic Append/Delete Controls

Automatically adds append/delete controls to (i) tables with a repeating table structure, and (ii) rows in a table with a repeating row structure. The **`Append`** button *(see screenshot below)* appends another occurrence of the repeating structure. The **`Delete`** button is available for each occurrence of a repeating structure, and deletes that occurrence. The screenshot below shows these buttons in the repeating table structure of a solution.



| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

This property defines whether the **`Append`** and **`Delete`** buttons are automatically added to the design. The property can have values of `true` and `false`. The default is **`true`**.

▼ Create For Each Item In

This property is available for tables, row groups, and column groups. It sets the number of times the item (table, row, or column) repeats to the number of items returned by the property's XPath expression.

The XPath expression can select a repeating node in a data tree. This is useful if you want the table, row, or column to be generated as many times as a certain node occurs and want to set the number of these repeats dynamically. In this way, if the number of the selected node's occurrences changes, then the number of repeats is automatically modified. For example, if the property's XPath expression is `$XML1/Office/Department`, then the number of times that the table, row, or column is generated will be equal to the number of `$XML1/Office/Department` elements.

The XPath expression can also be unrelated to a data tree. In this event, the table is created once for each item in the returned sequence. For example:

- 9 times for the expression `1 to 9`
- 2 times for `"John", "Mary"`
- 2 times for `45, true()`

▼ Max Table Width

The **`Max Table Width`** property specifies the table's width: (i) in pixels, dp, or sp, (ii) relative to the device's screen width, or (iii) optimized for columns (`wrap_content`). The default is `wrap_content`. Select the value you want from the dropdown list of the property's combo box. If the table width is larger than the screen width, then the table will be displayed with a horizontal scroll bar.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

For more information, see Table Properties 1078.

▼ Max Table Height

The **`Max Table Height`**[614] property specifies the table's height on the device's screen, in pixels, dp, or sp, or relative to the device's screen height. Select one of the values from the property's combo box. For example, if you select `50%`, then the table will have a height of half of the device's screen height. If the table has a vertical extent that does not fit in the allotted screen space, then the table will have a vertical scroll bar and the end user can scroll the rest of the table in and out of the allotted screen space (in this example, 50% of the screen height). If design components occur above the table, then all these components are displayed above the table; the table itself will have the absolute or relative height specified via this property.

Besides pixel values and percentages of the screen height, the **`Max Table Height`** property can take two other values:

- *Rest of screen height (max):* The table height is minimized as much as possible so that as much of the rest of the page (above and below the table) can be displayed. If, while you scroll down a page, you want the rows of the table to scroll but the content above it to stay fixed, then use this property value.
- *Rest of screen height (always):* This option enables you to use the entire screen height to display the page. If a table does not have enough vertical extent to fill the page, then additional space is added below the table so that the last component of the page is displayed just above the bottom of the screen.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

For more information, see Table Properties[1078].

▼ Scroll Vertically

The **`Scroll Vertically`** property becomes available after a value has been set for the **`Max Table Height`** property. It can take one of two values:

- *Whole table:* The whole table scrolls in and out of the table height that is allotted to the table via the **`Max Table Height`** 614 property. *Whole table* is the default value.

- *Rows except header and footer:* The table's header and footer stay fixed in the view. The table's body rows scroll in the remaining table height.

For more information, see Table Properties 1078.

### Row Group Chunk Size

The **`Row Group Chunk Size`** property becomes available only if there is a repeating row-group in the table and after a value has been set for the **`Max Table Height`** property of scrollable tables 1085. It enables you to specify the number of row groups that are loaded at a time. When the user scrolls down and the last row group of the last-loaded chunk is reached, then the next chunk is loaded. There is no default value for this property.

For more information, see Table Properties 1078.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

### Wrap Content Auto-Fit Group

You can select different controls, each of which is in a table cell, to belong to a group. Do this via the control's `Text Size Auto-Fit` property. After you have grouped controls in this way, you can auto-size the text strings in cells of a group so that: (i) all text is the same size, (ii) each text string fits in its respective cell, and (iii) all columns of the group are displayed at an optimum width. In effect, the text size and column widths are optimized so that all of the text in all columns is visible within the available display area. Note that a column will be re-sized only if its **`Width`** property has been set to `wrap_content`; if this is not the case, then only the text size will be adjusted while the column remains relatively large.

Note that this optimization is time consuming and should typically be used only on smaller tables.

The property is available on tables. It's values may be any `Text Size Auto-Fit` group that has been used to identify a set of cells or a template group. The content of the selected group is auto-sized, and the table cells are wrapped within the available width (instead of being cut off). This enables even long rows to be shown within the available width. The property's default value is **`off`**.

For more information, see the `Text Size Auto-Fit` property of any control that features text (such as the Button 417 or Label 554 control).

### Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Border Width

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the `Border Width` property to display the individual border width properties (left, right, top and bottom), and set the different value. For example: If you set `Border Width` to be `1px` and `Border Width Top` to be `2px`, then the widths of the left, right and bottom borders will all be 1px, while the top border width will be 2px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

▼ Border Color

Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the Border Color property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set Border Color to be blue and Border Color Top to be red, then the color of the left, right and bottom borders will all be blue, while the top border color will be red.

▼ Border Style

Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four sides, expand the Border Style property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set Border Style to be dashed and Border Style Top to be solid, then the style of the left, right and bottom borders will all be dashed, while the top border style will be solid.

▼ Apply Borders to Cells

Select a value of true to apply the table's border properties to the borders of all cells in the table. (The default value of the property is false.) This is a useful property if you wish to apply border properties to all cells at a stroke.

Note the following points:

- The properties will be applied only to those table cells that do not have any border property assigned to them.

- The properties will be applied only to the top and left borders of cells.
- If this property is set to `true`, then changes to a table's border shorthand property (except for right and bottom border properties) will also be passed to the cell borders.
- Changes to individual table borders (top, right, bottom, or left) will not be passed to cell borders.

### Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

### Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

### *Note about the properties of nested tables*

If an **entire** nested table is selected, then: (i) properties of the nested table are listed under the heading *Table*, while (ii) properties of the parent table's containing cell, column, and row are listed, respectively, under the following headings: *Parent Table's Cell*, *Parent Table's Column*, and *Parent Table's Row*. Note that nested tables can be horizontally and vertically aligned.

**Note:** Table borders can be conveniently set in the Border Settings dialog (**Table | Border Settings**[1644]).

## 9.1.22 Time

Time controls are used to format times obtained from a node in a page source. This is useful if you wish to format times in a special way. The formatting is defined in the `Date/Time Format String` property (*see below for details*). Note that the source node content must be in the correct lexical format as defined by the XSD specification: `HH:MM:SS`. Optional timezone and millisecond parts are allowed.

### Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.

- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

## Time (control) events

The `OnFinishEditing event`[665] is available. For a description of the actions that can be defined for this event, see the Actions section[667].

## Time properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### ▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### ▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }
```

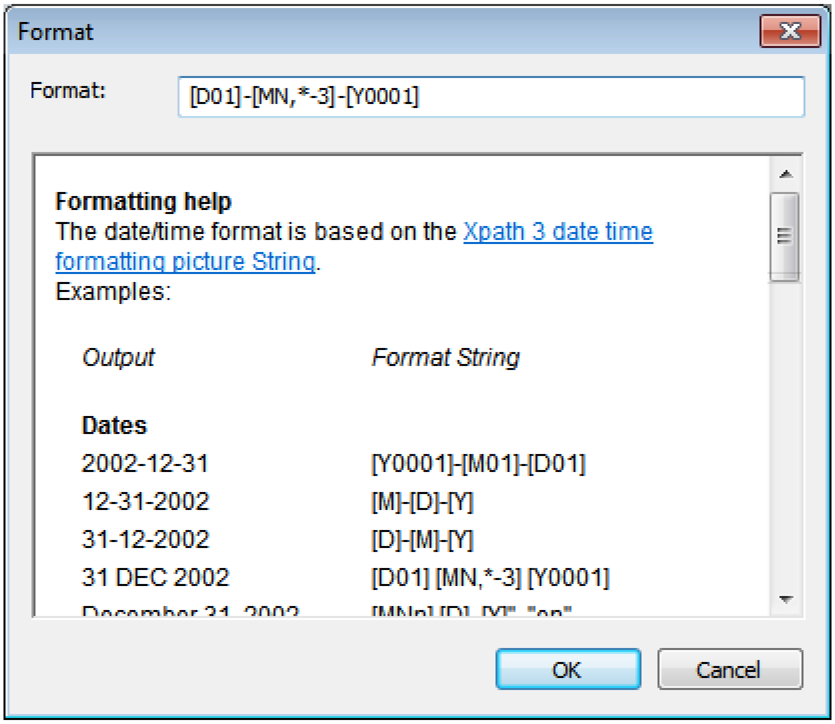
```
map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane 274 will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane 274.
- You can also specify a style sheet to use, as shown in the second map above.

### Date/Time Format String

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (*screenshot below*).



Format: [D01]-[MN,*-3]-[Y0001]

**Formatting help**
The date/time format is based on the Xpath 3 date time formatting picture String.
Examples:

| *Output* | *Format String* |
|---|---|
| **Dates** | |
| 2002-12-31 | [Y0001]-[M01]-[D01] |
| 12-31-2002 | [M]-[D]-[Y] |
| 31-12-2002 | [D]-[M]-[Y] |
| 31 DEC 2002 | [D01] [MN,*-3] [Y0001] |
| December 31, 2002 | [MNn] [D], [Y]", "en" |

The formatting will be applied to the control's content if the content has the correct lexical form of `xs:date` (for the Date control), `xs:time` (for the Time control), or `xs:dateTime` (for the Date, Time, and DateTime controls). Basic examples are:

- *xs:date:* `2014-12-31`
- *xs:time:* `23:59:59`

- *xs:dateTime:* `2014-12-31T23:59:59`

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is `true` (enabled) or `false` (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean `true` or `false`. The default value is `true`. Typically the XPath expression of the `Enabled/Editable` property of a control would check the presence or value of a node. If the expression evaluates to `true`, the control will be enabled.

▼ Assertion

Sets a condition to be met for the control to be valid. If the assertion is invalid, then the text of the `Assertion Message` property (*see below*) is displayed in the Assertion Message 409 control. (If there are multiple Assertion Message 409 controls, then all these controls will display the text of the `Assertion Message` property.) Click the property's **XPath** icon to enter an XPath expression that defines the assertion. For example: The XPath expression `LastName != ""` asserts that the node `LastName` must not be empty. If this node is empty, then the control's assertion message is displayed in the Assertion Message 409 control of the page.

Note that other controls and the page can also have assertions. If there are multiple invalid assertions on a page, then the assertion message of the first invalid assertion is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

### Assertion Message

Sets the assertion message to be displayed if the control's assertion is not valid. Double-click inside the value field of the property to edit the assertion message, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. The assertion message is displayed in the Assertion Message[409] control. For example: If the XPath expression of a control's assertion is `LastName != ""`, then it asserts that the node `LastName` must not be empty. If this node is empty, then the assertion message of the control is displayed in the Assertion Message[409] control of the page.

Note that assertions can also be defined for other controls and the page. So it can happen that there are multiple invalid assertions on a page. If this happens, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. Control assertions are evaluated before page assertions, and control assertions are evaluated in the order in which they occur in the design.

### Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is disabled, use the **`Text Color (Disabled)`** property.

### Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a text color for when the control is enabled, use the **`Text Color`** property.

### Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest|small|medium|large|largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` 1262 function. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression for the **`TextSize`** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's values are:

- `off` *(the text is not automatically resized; this is the default)*
- `ellipsis` *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- `individually` *(the text of this control only will be automatically resized)*
- `group X` *(where X=1 to 9).* You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- `template group X` *(where X=1 to 9).* Like for a group (see previous list item), you can set a control in a control template 1200 to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A *template group* is different from a *group* (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a *template group* as a *group* if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

*Note*

- If the `Multiline` property has been set to **`true`**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the `Max Number of Lines` label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's `Wrap Content Auto-Fit Group` property.

### ▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is `false`.

### ▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is `false`.

### ▼ Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

### ▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box. If the cell contains a control, you might want to select the option *Control's background color*.
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **`Background Color (Disabled)`** property.

### ▼ Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is enabled, use the **`Background Color`** property.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Hint

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Text Color Hint

Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (*see above*). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304]), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

### Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

#### *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the `Padding` property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set `Padding` to be `6px` and `Padding Bottom` to be `12px`, then the top, left and right padding will be `6px` and the bottom padding will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Tab Order

The `Tab Order` property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** 1633 menu command. The `Tab Order` property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu 404.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control 1318. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets* 1327). See the section Style Sheets 1318 for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog 296) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property 384.

## 9.1.23 Vertical Line

The Vertical Line control enables you to add vertical lines within tables. You can style vertical lines for properties such as color and width; the available properties are listed below. If you copy a vertical line to another location using **Ctrl**+drag-and-drop, note that you can copy only to a location inside a table.

⊟ Notes

- To reset a style or property (in the Styles & Properties Pane 274), select the property and click **Reset** in the pane's toolbar 276.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane 274), select the style or property, and click **Edit XPath** in the pane's toolbar 276.
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.

### Vertical Line events

There is no event associated with the Vertical Line control.

## Vertical Line properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

### Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

### All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

### Line Width

Sets the width (thickness) in pixels of the line. Select a width value (in pixels, dp, or sp) from the dropdown list of the combo box, or double-click in the value field to enter a numeric value. You can also use an XPath expression. The numeric value is understood to be a pixel value. For this reason, no unit should be specified.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Line Style

Specifies the style of the line. You can select one of the options from the dropdown list of the combo box, or use an XPath expression. The default value is `solid`.

▼ Line Color

Specifies the color of the line. You can do one of the following to select the color:

- Click the color palette to select a line color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate or fetch the required color code

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a `Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu[404].

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

▼ Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.1.24 Video

The Video control displays the video that is specified in the control's `Video Source` property (*see below*). Multiple video controls can be placed on a page. Each control is identified by a name and plays the video specified in its `Video Source` property. Properties of the control can be set in the Styles & Properties Pane[274].

Note the following features:

- Video resources are located via URLs. The URLs can be specified via a page source link[347], an XPath expression, or a directly entered static address. To specify a video file on the client device, the URL must be a relative URL that is relative to the respective Device Dependent Directory (*see the description of the* `Video Source` *property below*). For information on video file formats, see Audio/Video Formats[1115].
- You can specify the control's size for the time during which the video is being downloaded. After the download has been completed, the control expands or contracts to the video's actual size. See the `Initial Width` and `Initial Height` properties below.
- The video can be started when the page loads (`Play on Load` property). If video playback is to be started at a later time, use the Video Start action[728] (for example, on a Button[417]).
- You can set whether buttons that control the video's playback actions are displayed in the control or not (`Show Controls` property). If you prefer not to display these buttons, the Video action[728] can be use to create custom buttons.

- The Video action[728] provides functionality to start, pause, resume, stop, and skip to a specific time in the playback, as well as to play a time-defined segment.
- You can also specify a set of actions to perform when a video event[656] is triggered (*see below*[656]).
- A number of MobileTogether XPath extension functions[1262] that provide attributes of video files and playback can be used to build conditional processing. For example, you can define alternative sets of actions to carry out if the **`mt-video-is-playing`** [1262]function returns `true()` or `false()`.

⊟ Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View[253]) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View[253]) and click **Unassign Page Source Link <NodeName>**.
- To reset a style or property (in the Styles & Properties Pane[274]), select the property and click **Reset** in the pane's toolbar[276].
- The values of several properties can be set by using XPath expressions. This allows values to be dynamic: that is, generated via calculations, or from data source nodes, at runtime. To set an XPath expression, click **Edit XPath** in the toolbar of the Styles & Properties Pane[276].
- To edit the XPath expression of a style or property (in the Styles & Properties Pane[274]), select the style or property, and click **Edit XPath** in the pane's toolbar[276].
- **To copy a control** to another location in the design, press **Ctrl** and drag-and-drop the control to the desired copy location.
- To assign specific properties for a control, define one or more classes for the control (via the Browser CSS Class property), and then assign rules for the class/es in an external CSS file (that you specify in the Browser Settings dialog[296]).
- A control's CSS properties can be defined in the Styles & Properties Pane[274] and/or in an external CSS file[296]. Those defined in the Styles & Properties Pane[274] have priority.

## Video events

Video events are accessed from the control's context menu (right-click to open) or the control's `Control Action` property (*see properties below*). To define actions for a video event, drag and drop an action from the left-hand Actions pane into the video event's tab.

- **`OnVideoStarted`**: Before this event occurs (that is, before the video starts playing), **details of the video file are not available**, and the functions to get video height, width, duration, and current position (*see below*) should not be called; at this time, only the **`mt-video-is-playing`** function will return valid information. This event can be used, for example, to log details of video playback (say, via the Update Node action[886]) to an XML tree node.
- **`OnVideoError`**: Possible errors could be: File not Found, a file format error, or download/playback interruption. Information about the error can be retrieved with the MobileTogether XPath extension function `mt-external-error`[1262]. If actions are defined for the event, these actions are executed. Otherwise, the error is shown in a message box.
- **`OnVideoCompleted`**: Video playback is considered to be completed when the file or specified segment plays to the end (without a Stop action being executed). The actions defined for this event are **not** performed when the video is suspended (with the project property *On Switch to Other Solution*[296]*)* or paused.

## Video properties

The control's properties are available in the Styles & Properties Pane[274], and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ All Styles

The `All Styles` property becomes visible if the *All Styles* setting of the More Project Settings dialog[296] has been set to `true`. (The default of this setting is `false`.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

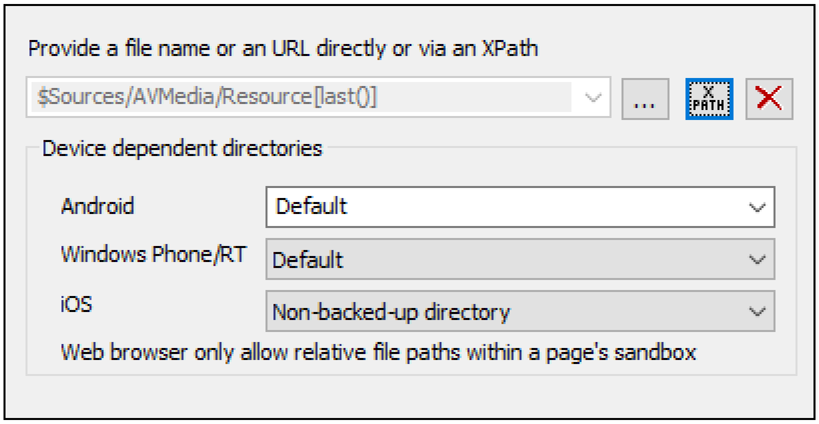
```
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
   }

map{
    "Style Sheet"      : "Sheet-1"
   }
```

Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane[274] will no longer be visible .
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **`Bold Text`** and **`Text Size`** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane[274].
- You can also specify a style sheet to use, as shown in the second map above.

▼ Video Source

Selects the video source to play in the video control. You can enter a URL that locates a remote file, or enter an XPath expression that selects or generates the URL. In the screenshot below, for example, the URL contained in the last **`$Sources/AVMedia/Resource`** element is used to locate the video file to play.



Provide a file name or an URL directly or via an XPath

$Sources/AVMedia/Resource[last()]

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

To specify a video file on the client device, the URL must be a relative URL that is relative to the respective Device Dependent Directory (*see screenshot above*).

- *Android:* Select the Android device-directory from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. Note, however, that, unless the Android device is rooted, no other app (besides MobileTogether) will be able to access the MobileTogether sandbox directory. So, trying to open a file in the MobileTogether sandbox with another app could fail.
- *Windows RT:* Select the Windows Phone or Windows RT device folder from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected.
- *iOS:* MobileTogether creates two directories on the user's device: (i) a *Backed-up directory* for files that are backed up by the operating system and will be available at later times, for example, after a restore. This directory is intended for files that are important for the user and that should not be lost; (ii) a *Non-backed-up directory* for files that do not need to be backed up, or if faster buffering is needed for playback. Select *Backed-up directory* or *Non-backed-up directory* as required.
- *Web browser:* No selection is available. Relative paths are resolved within the context of the browser's sandbox.

For information on video file formats, see Audio/Video Formats 1115.

**Note:** Multi-channel audio/video playback is not supported on Windows Phone. Only one audio **or** video file can be played at a time: this is the file that was started last.

**Note:** Audio and video files **cannot** be deployed to MobileTogether Server via the MobileTogether Designer project's Deploy to Server mechanism 291. You can, however, copy audio/video files manually to the server, although you cannot stream them from there via a URL. If you wish to stream audio/video files that are located on your MobileTogether Server, then do the following: (i) use the Load Binary 815 action to load the binary audio/video data to a page source node; (ii) use the Save Binary 815 action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 1108 to play the file that is now saved on the client device. Alternatively, you can save audio/video files to a web server—instead of saving to MobileTogether Server—and use a URL to stream the audio/video file from the web server.

▼ Cached Video Source

If the property **`Play on Load`** (*see below*) is set to `false`, then you can specify a cache file from which to play the video. If a cache file is specified, then the video file selected for playback (via the **`Video Source`** property) will be cached on the client device when the video file is downloaded. If a local cache file already exists for a given video control, then the cache file will be played in the control and no fresh download takes place.

To specify a local cache file, enter a relative URL as the value of this property. This URL will be resolved relative to the folder selected for each OS in the *Device Dependent Directories* pane. You can specify whether sub-folders of the relative URL should be created if they do not exist on the client device.

- *Android:* Select the Android device-directory from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. Note, however, that, unless the Android device is rooted, no other app (besides MobileTogether) will be able to access the MobileTogether sandbox directory. So, trying to open a file in the MobileTogether sandbox with another app could fail.
- *Windows RT:* Select the Windows Phone or Windows RT device folder from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected.
- *iOS:* MobileTogether creates two directories on the user's device: (i) a *Backed-up directory* for files that are backed up by the operating system and will be available at later times, for example, after a restore. This directory is intended for files that are important for the user and that should not be lost; (ii) a *Non-backed-up directory* for files that do not need to be backed up, or if faster buffering is needed for playback. Select *Backed-up directory* or *Non-backed-up directory* as required.
- *Web browser:* No selection is available. Relative paths are resolved within the context of the browser's sandbox.

▼ Username

This property is enabled if `Image Source Type` is `url`. Sets a user name for user access to the resource. Double-click in the property's value field to edit.

▼ Password

This property is enabled if `Image Source Type` is `url`. Sets a password for user access to the resource. Double-click in the property's value field to edit.

▼ Show Controls

Specifies whether the buttons of the video control are displayed or not. The values of the property are `true` or `false`, with the default being `true`. If the value is set to `false`, playback can be controlled via the Video action 728. Note that Video control buttons are not supported on Windows Phone.

▼ Play on Load

Specifies whether the video plays directly after the page has been loaded. The values of the property are `true` or `false`. The default value is `true`. If the value is set to `false`, playback is started with the Video Start action 728.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog 667. You can set actions to perform when a control event 665 is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog 667. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties 1078.

**Note:** The `$MTControlValue` 1304 variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

▼ Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to `left`, `center`, or `right`. Default is `left` for all controls except vertical lines, for which it is `center`. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ Vertical Alignment

Sets the vertical alignment to `top`, `middle`, or `bottom`. Default is `middle`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the `Multiline` property)

▼ Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available

- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** 1304 and **`$MT_CanvasY`** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Max Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a

value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Initial Width

A value in pixels, dp, or sp that specifies the initial width of the video control. This is the width of the video control while the video is downloading. When the video starts playing, the control's width expands or contracts to the actual width of the video. The default initial width is the value of **`$MT_CanvasX`**[1304].

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Initial Height

A value in pixels, dp, or sp that specifies the initial height of the video control. This is the height of the video control while the video is downloading. When the video starts playing, the control expands or contracts to the actual height of the video. The default initial height is **`($MT_CanvasX`**[1304] **`* 9) div 16`**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the `Margin` property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set `Margin` to be `6px` and `Margin Bottom` to be `12px`, then the top, left and right margins will be `6px` and the bottom margin will be `12px`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*[1312].

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **`$MT_CanvasX`**[1304] and **`$MT_CanvasY`**[1304] dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Variables

Opens the control's Local Variables dialog, where you can add, edit, and delete the variables of the control. A control variable has a name, which is a string, and a value, which is an XPath expression. The variables that are declared in the Local Variables dialog will be evaluated when the control is called and will be used with this value till the control is called again. Parameters can be used in the XPath expressions that calculate the variable's value.

Control variables are useful for providing values that can be used in control actions. For example, a control variable can select a domain name based on the control's context. So, if a the current node is, say, a

`Department` node inside a `Company` node, then the current company could have a domain name of, say, `altova.com` or `nanonull.com`. With the domain name set as a control variable, the control variable can then be used in an action of the control. For example, email addresses of a department's `Employee` children can be built by using each employee's `FirstName` and `LastName` elements together with the control variable that contains the company's domain name.

Control variables can also be set via a control's context menu[404].

### Style Sheet

The `Style Sheet` property sets the style sheet to use for the control[1318]. The dropdown list of the `Style Sheet` property's combo box displays all the user-created style sheets that have been defined in the project. Select the style sheet you want to use for the control. Alternatively, you can use an XPath expression to select the style sheet you want; this has the advantage that you can make the selection of the style sheet conditional (*see Applying User-Created Style Sheets*[1327]). See the section Style Sheets[1318] for more information.

### Browser CSS Class

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog[296]) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

You can use the following predefined value to set specific behavior:

- **`mt-combo-open-on-focus`** opens the dropdown list of a combo box when the user tabs to it. You can use this value on (i) combo box controls and (ii) on table controls if the table contains combo box controls. Alternatively, if you want to open all combo boxes on a page individually, then you can set this value on the `Browser CSS Class` page property[384].

## 9.2 Control Events

Most of the controls used in page designs have predefined **events** associated with them (*see table below*). For example a button control has an `OnButtonClicked` event associated with it. These events are called **control events**, and each can have an **action (a control action)** 667 defined for it. When a control event is triggered while a MobileTogether solution is being executed, the control action that has been defined for the event is carried out.

The table below lists controls and their events.

| Control | Event |
|---|---|
| Assertion Message 409 | – none – |
| Button 417 | `OnButtonClicked` |
| Chart 438 | `OnImageClicked` |
| Check box 447 | `OnFinishEditing` |
| Combo box 459 | `OnFinishEditing` |
| Date 473 | `OnFinishEditing` |
| DateTime (iOS) 484 | `OnFinishEditing` |
| Edit field 495 | `OnTyping` |
| Geolocation Map 508 | `OnGeoMapMarkerClicked` |
| Horizontal Line 515 | – none – |
| Horizontal Slider 520 | `OnSliding` |
| Image 541 | `OnImageClicked` |
| Label 554 | `OnLabelClicked` |
| Radio button 571 | `OnFinishEditing` |
| Rich Text 584 | `OnFinishEditing` |
| Signature field 590 | – none – |
| Space 600 | – none – |
| Switch 603 | `OnFinishEditing` |
| Table 614 | – none – |
| Time 640 | `OnFinishEditing` |
| Vertical Line 651 | – none – |
| Video 655 | `OnVideoStarted, OnVideoError, OnVideoCompleted` |

## Defining the actions of a control event

Action/s of a control event are defined in the Actions dialog 667 (*see screenshot below*). Drag the desired action from the left-hand pane into the event tab in the right-hand pane.

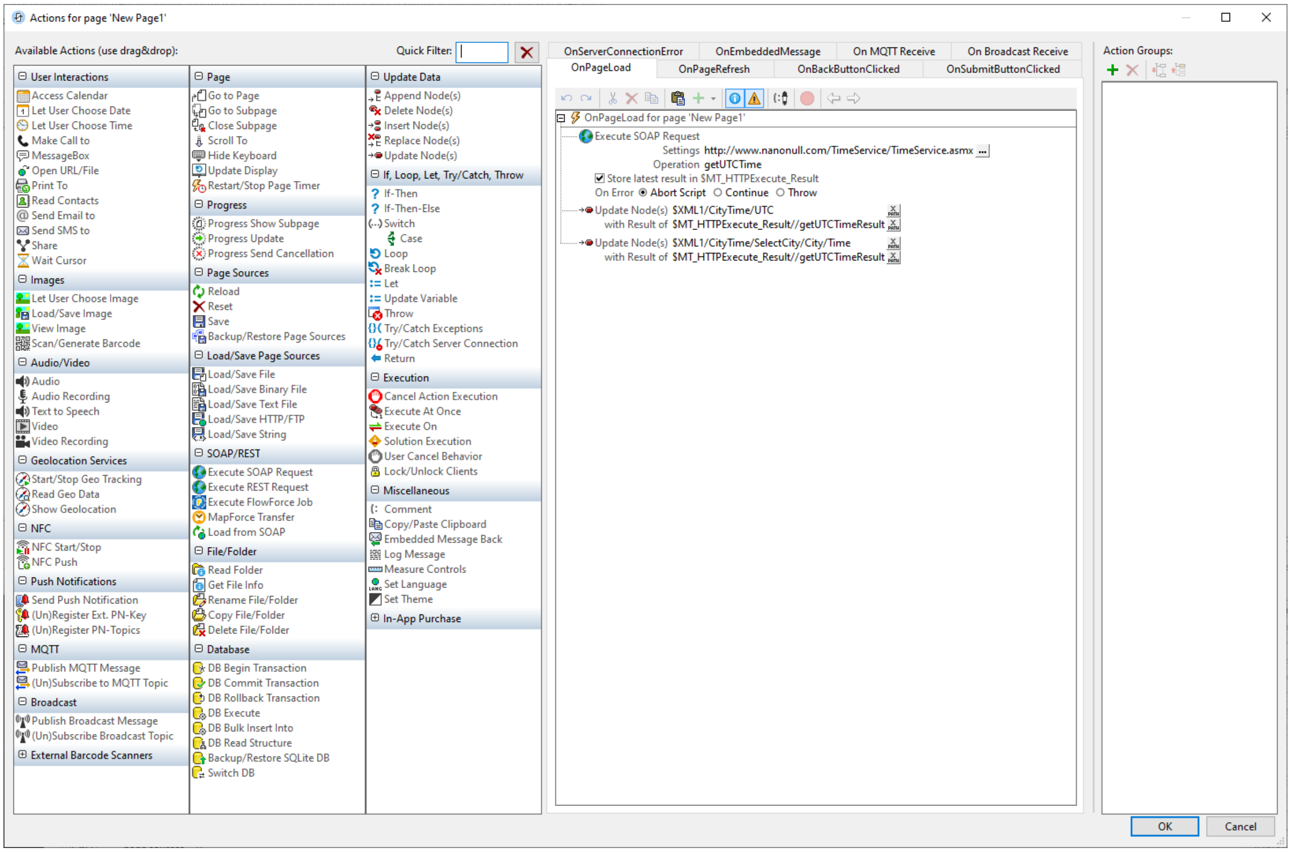
The Actions dialog 667 of a control is accessed in one of the following ways:

- Right-click the control, and select **Control Actions for ...**
- In the Styles & Properties Pane 274, click the **Additional Dialog** button of the `Control Action` property
- Click **Page | Actions Overview** 1632 to display the Actions Overview dialog 1632. Then click the **Additional Dialog** button of the control event you want to define.

For more information about the various types of available control actions, see the section Actions 667.

# 10 Actions

Actions can be defined for control events 665 and page events 389. Actions for both kinds of events are defined in the Actions dialog (*see screenshot below*). The left-hand pane contains the available actions grouped according to functionality. Each group can be expanded/collapsed so that infrequently used actions are hidden; this makes frequently used actions easier to find. The main pane in the center contains the events of the control or the page, and the right-hand pane is for managing Action Groups 940.



OnPageLoad for page 'New Page1'

Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx
Operation getUTCTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw

Update Node(s) $XML1/CityTime/UTC
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

Update Node(s) $XML1/CityTime/SelectCity/City/Time
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

In the central main pane, each event has a separate tab, in which you can define the actions to carry out for that event. Most controls typically have a single event, whereas a page has several events. The screenshot above shows the actions that have been defined for the `OnPageLoad` event of a page.

To define a certain action for the selected event, drag the action from the left-hand pane into the event tab. If an action requires further definitions—such as selecting an entry from a combo box, or entering an XPath expression (*see screenshot above*)—then complete these steps.

You can also define **multiple actions** for an event and add **child actions** to an action. In the screenshot above, for example, there are three actions defined for the `OnPageLoad` event. Click **OK** when you have finished defining the control or page action/s. If there are multiple actions on the same level, then they are performed in the order in which they are defined.

The following keyboard shortcuts are available:

| Shortcut | What's clicked in the event pane | Result |
|---|---|---|
| **Ctrl+double-click** event/action | Main event (e.g `OnLabelClicked`) | Action added as last action of the event |
| | Sub-event (e.g `OnClick`, `On LongClick`) | None |
| | Action | Double-clicked action added as next action |
| **Alt+double-click** event/action | Main event (e.g `OnLabelClicked`) | Action added as last action of first sub-event |
| | Sub-event (e.g `OnClick`, `On LongClick`) | Action added as last action of sub-event |
| | Action | None |

## Inserting actions from a popup

The conventional way of inserting an action for processing when an event is triggered is as described above: to drag the action from the Available Actions pane (on the left) to the event pane (center). However, you can also insert an action by selecting it from a popup in the event pane.

Insert an action from a popup as follows:

1. Press **Alt** on your keyboard.
2. Move the mouse cursor to the location where you want to insert the action. The cursor becomes a horizontal line that indicates where the action can be inserted.
3. When the cursor is placed over the insertion location you want, click the mouse. A popup appears that contains an edit field.
4. Enter (i) the first few characters of the name of the action (or action group) you want to insert, or (ii) a text string that is contained in the action's name. The best five matches are shown in the popup.
5. In the popup, select the action (or action group) you want, either (i) by clicking it, or (ii) by using the **Up** and **Down** keys to select the action and then pressing **Enter**. The selected action will be inserted at the insertion location.

## Filtering actions and action groups by name

To filter actions and action groups by name, enter a part of an action or action group's name in the *Quick Filter* text box at the top right of the Available Actions pane (*see screenshot above*). The matched actions and/or action groups will be highlighted in green. Matched action groups will be shown at the top of the Action Groups list, and they will be sorted alphabetically.

The filter feature is useful if you wish to see actions of a particular type. For example, if you wish to display actions related to the saving of data, you could enter `Save` in the *Quick Filter* text box.

## List usages of an action or action group

Right-click an action or action group in the Available Actions pane, and, in the context menu that appears, click **List Usages of this Action**. All usages of that action in the design will be displayed in the Listings pane 281. The usages are listed by page, and then by control.

## Order of action execution affects performance

Some actions are executed on the server (generating charts, loading non-embedded files, etc) and some are executed on the client (message boxes, sending an SMS, etc). Therefore, in order to improve performance, the order of actions should be defined in such a way that the switching between server and client for processing is minimized.

## Disabling an action

You can temporarily disable any individual action that is defined for an event. A disabled action is ignored, and processing continues as if the disabled action is not defined. To disable an action, right-click it in the definition of the event's actions and select **Disable Action**. This command is a toggle command, so selecting it once more will enable the action again.

## Aborting actions on errors

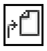
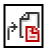
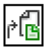
You can set the project to abort actions when errors occur during action handling. The setting to do this is in the More Project Settings dialog[296]. By default, the *Abort* option is set to `true`. The error that triggers the action abortion could be in an XPath expression or elsewhere in the action handling. However, minor errors such as XPath errors for selecting a style property are ignored and action handling continues.

## Suppressing logs recursively

Logs are the messages that are displayed in the Messages Pane[278]. The messages relating to an individual action and its sub-actions (child actions) can be suppressed. Do this by right-clicking the action and selecting **Suppress Logs Recursively**.

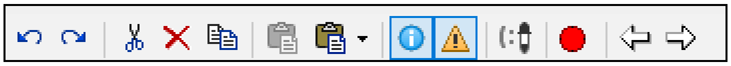
If you want to activate the logs of a child action of an action that has had its logs suppressed recursively, then right-click the child action and select **Force Enabled Logs Recursively**.

Actions for which logs can be suppressed and force-enabled have icon overlays that indicate the current status. The screenshots below show these icon overlays of the Go to Page[783] action.

| | |
|---|---|
|  | *No overlay* |
|  | *Logs suppressed* |
|  | *Logs force-enabled* |

## Event pane toolbar

The Event Pane toolbar (*screenshot below*) offers the following commands, starting from the left:



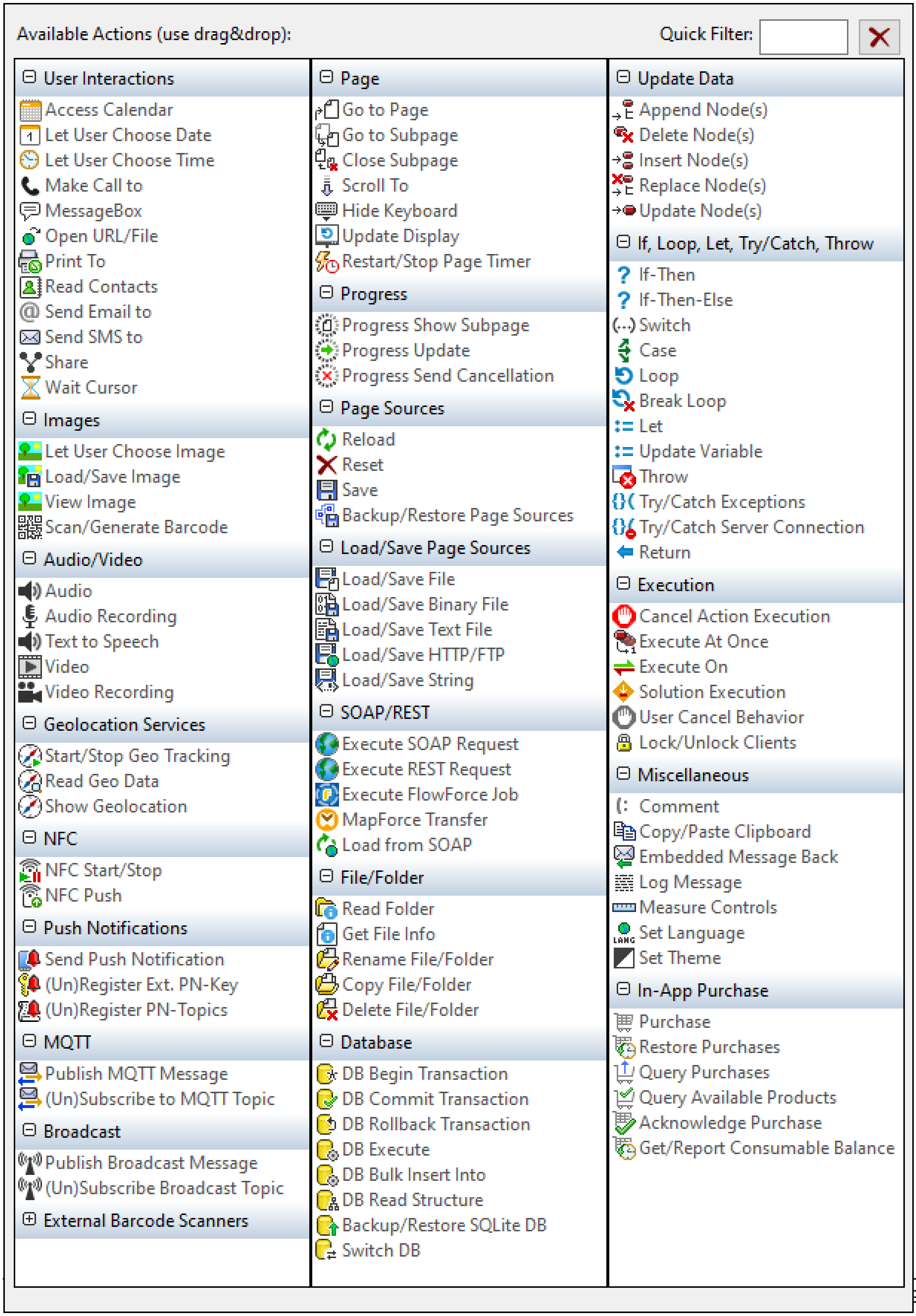
- *Undo, Redo:* Undoes and redoes your Event Pane edits
- *Cut, Delete:* Deletes the selected item in the Event Pane. *Cut* additionally puts the selected item on the clipboard
- *Copy:* Copies the selected item to the clipboard

- *Paste, Append Clipboard Contents:* Pastes the clipboard contents relative to the selected location
- *Show/Hide Rarely Used Options:* Some actions have options (or settings) that are inessential to the working of the action. This setting toggles on/off the display of such options
- *Show/Hide Warnings for Actions:* If some mandatory condition required for an action to work correctly is missing, then a warning is displayed in red. This setting toggles on/off the display of warnings
- *Comment Color:* Displays a color picker that lets you set the color of text in the Comment action 923. The selected color will be applied to all comments in all designs in MobileTogether Designer, including to comments that were defined prior to the setting of the new color
- *Toggle Breakpoint (F9):* Toggles a breakpoint on/off the selected action
- *Back, Forward*: Cycles through the design's Action Groups that were opened in the current session.

# 10.1 User Interactions

The following actions are available in the User Interactions group of the Actions dialog (*screenshot below*):

- Access Calendar 673
- Let User Choose Date 677
- Let User Choose Time 678
- Make Call To 679
- Message Box 679
- Open URL/File 681
- Print To 686
- Read Contacts 692
- Send Email To 693
- Send SMS To 698
- Share 699
- Wait Cursor



Available Actions (use drag&drop): Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

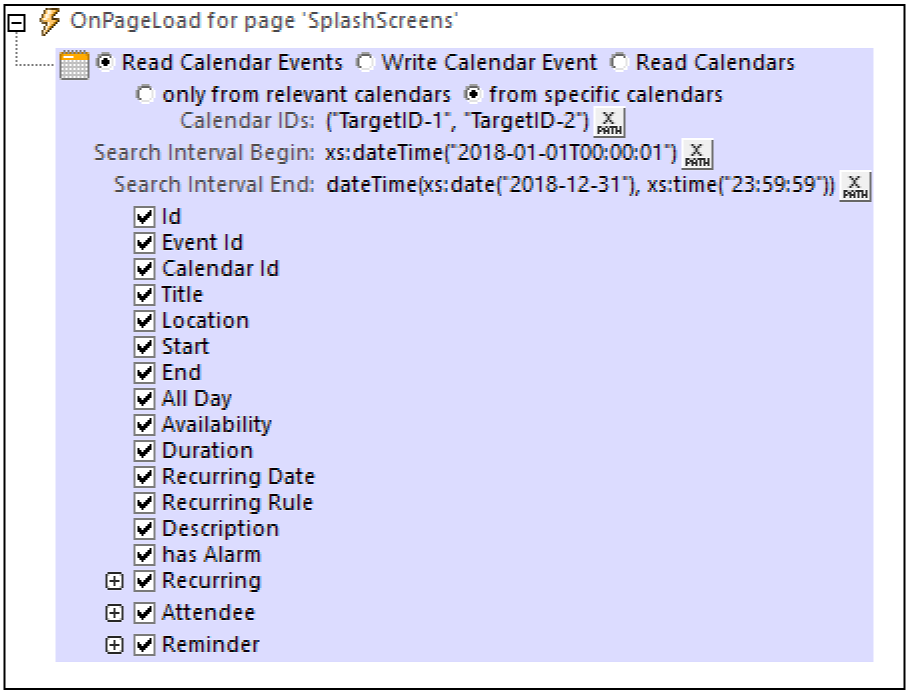
⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events* 389 *and Control Events* 665.

## 10.1.1 Access Calendar

When an Access Calendar action (*screenshot below*) is added to the design, the `$MT_CALENDAR` page source tree is automatically added to the design. At run time, depending on what kind of calendar action was selected, either (i) information from the device's calendars are read and stored in the `$MT_CALENDAR` tree, or (ii) an entry for a calendar event is opened in the device's calendar app; the user can edit this entry and then save it.

**Note:** Windows 8 client devices do not support calendar events.



OnPageLoad for page 'SplashScreens'

● Read Calendar Events ○ Write Calendar Event ○ Read Calendars

○ only from relevant calendars ● from specific calendars

Calendar IDs: ("TargetID-1", "TargetID-2")

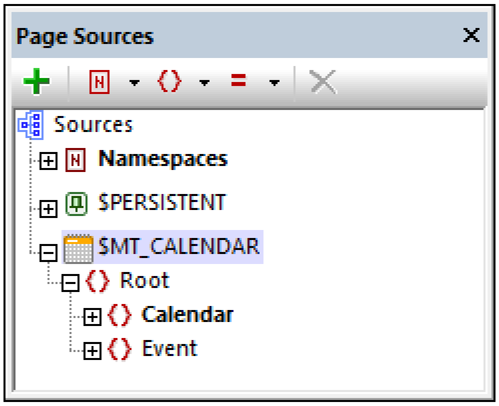
Search Interval Begin: xs:dateTime("2018-01-01T00:00:01")

Search Interval End: dateTime(xs:date("2018-12-31"), xs:time("23:59:59"))

☑ Id
☑ Event Id
☑ Calendar Id
☑ Title
☑ Location
☑ Start
☑ End
☑ All Day
☑ Availability
☑ Duration
☑ Recurring Date
☑ Recurring Rule
☑ Description
☑ has Alarm
⊞ ☑ Recurring
⊞ ☑ Attendee
⊞ ☑ Reminder

There are three kinds of calendar actions:

- *Read Calendar Events:* Reads event information in the calendars on the device, and saves this information to the `$MT_CALENDAR` tree. Each event is saved as a separate **`Event`** element (*see screenshot of the* `$MT_CALENDAR` *tree structure below*). Data in the tree can subsequently be used in the solution.



Page Sources

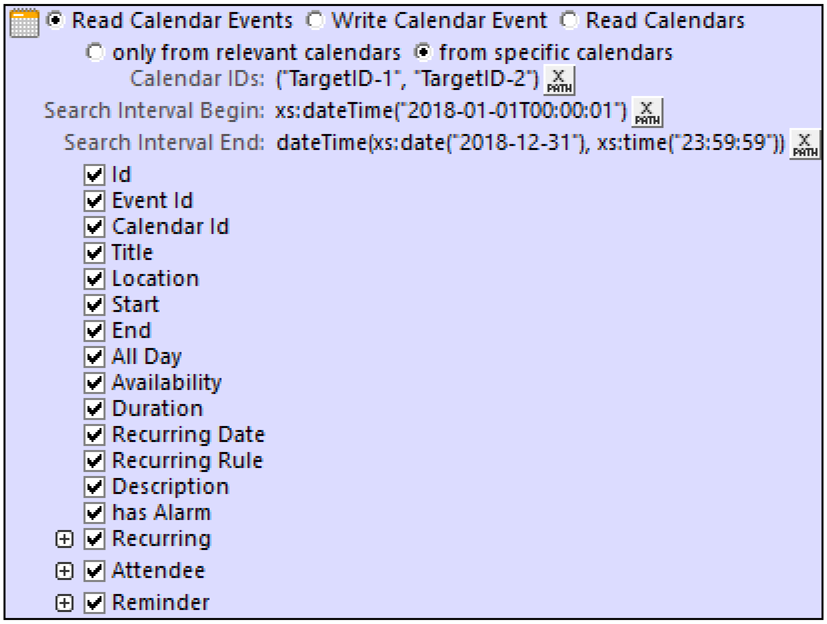
- Sources
  - Namespaces
  - $PERSISTENT
  - $MT_CALENDAR
    - Root
      - Calendar
      - Event

- *Write Calendar Event:* At run time, opens an event entry in the calendar app of the end user's device. This event entry will be filled with information that you entered in the action's settings. The end user can now edit and save the entry to the device's calendar.
- *Read Calendars:* Reads information about the calendars on the device, and saves this information to the `$MT_CALENDAR` tree. Each calendar is saved as a separate **`Calendar`** element (*see screenshot of the* `$MT_CALENDAR` *tree structure below*). Data in the tree can subsequently be used in the solution.

These three kinds of calendar action are described in more detail below.

## Read Calendar Events

This action (*see screenshot below*) reads information about events in the calendars on the device. You can select only the relevant calendars on the device (*see Read Calendars* 675 *above*), or you can specify calendars to read by their IDs (with multiple IDs being given as a sequence of strings). Each event is stored as an **`Event`** element in the `$MT_CALENDAR` tree.



● Read Calendar Events ○ Write Calendar Event ○ Read Calendars

○ only from relevant calendars ● from specific calendars

Calendar IDs: ("TargetID-1", "TargetID-2")

Search Interval Begin: xs:dateTime("2018-01-01T00:00:01")

Search Interval End: dateTime(xs:date("2018-12-31"), xs:time("23:59:59"))

☑ Id
☑ Event Id
☑ Calendar Id
☑ Title
☑ Location
☑ Start
☑ End
☑ All Day
☑ Availability
☑ Duration
☑ Recurring Date
☑ Recurring Rule
☑ Description
☑ has Alarm
⊞ ☑ Recurring
⊞ ☑ Attendee
⊞ ☑ Reminder

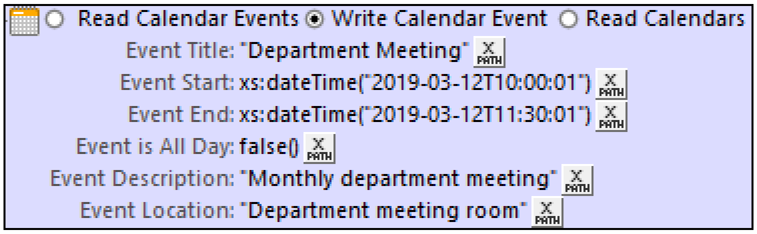
You can select what data fields of the event to read:

- *ID, Calendar ID:* A string that is the calendar's identifier.
- *Event ID:* A string that is the event's identifier.
- *Title:* The name of the event.
- *Location:* The venue of the event.
- *Start, End:* The start and end times of the event.
- *All Day:* Whether the event is to last all day. *All Day* is set if no start/end times are specified; if set this property has a value of **`true()`**, otherwise it has a value of **`false()`**.
- *Availability:* The availability of the calendar's user.
- *Duration:* Duration of the event in minutes.
- *Recurring Date, Recurring Rule:* The date on which the event recurs, and the recurrence rule (for example: weekly, on Thursdays).
- *Description:* A description of the event.
- *Has Alarm:* Whether the event has an alarm set: **`true`** for yes, **`false`** for no.
- *Recurring:* The start and end times of the period within which an event recurs.
- *Attendee:* The details of each attendee are stored in a separate *Attendee* item.
- *Reminder:* Details of the reminder, such as the reminder time interval and the method of the reminder.

**Note:** If the calendar contains no information for a particular field, then nothing is returned for that field.

## Write Calendar Event

When this action (*see screenshot below*) is executed, it opens the device's calendar app and creates an event entry containing the data that you entered in the action. For example, an event entry created by the action shown in the screenshot below will contain the event's title, start and end times, description, and event location. Notice that in the screenshot below the values for *Event is all day* must be one of the Boolean values, **`true()`** or **`false()`**.



○ Read Calendar Events ● Write Calendar Event ○ Read Calendars

Event Title: "Department Meeting"

Event Start: xs:dateTime("2019-03-12T10:00:01")

Event End: xs:dateTime("2019-03-12T11:30:01")

Event is All Day: false()

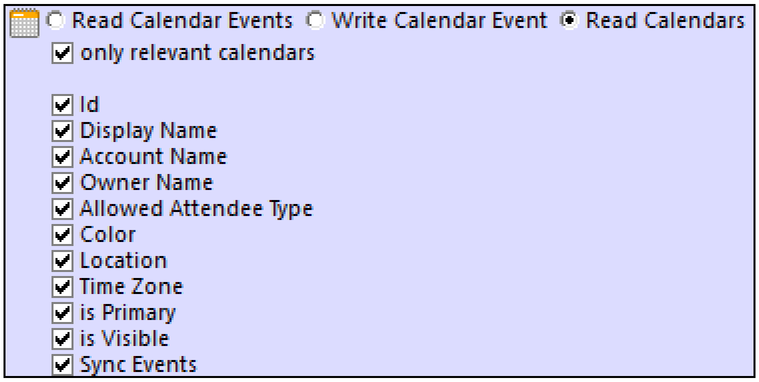
Event Description: "Monthly department meeting"

Event Location: "Department meeting room"

When the action is triggered on the client device, the event will not be saved directly to any calendar. Instead, the event entry will be opened in the calendar app so that the user can immediately edit it and save it to the calendar they want.

## Read Calendars

This action reads information about the calendars on the device. A device might have additional calendars, such as one for international holidays or for the trade fairs of a particular industry. These calendars are not normally used to add new events, and so are considered non-essential. When reading calendars, you can filter out these "non-essential" calendars (by selecting *Only Relevant Calendars; see screenshot below*). If the calendar selection is unfiltered, then all calendars on the device are read. Each calendar is stored as a **`Calendar`** element in the `$MT_CALENDAR` tree.



○ Read Calendar Events ○ Write Calendar Event ● Read Calendars

☑ only relevant calendars

☑ Id
☑ Display Name
☑ Account Name
☑ Owner Name
☑ Allowed Attendee Type
☑ Color
☑ Location
☑ Time Zone
☑ is Primary
☑ is Visible
☑ Sync Events

You can select what data fields of the calendar information to read:

- *ID:* A string that is the calendar's identifier.
- *Names:* These names can be used to disambiguate among calendars. Select one or more from among the calendar's display name, account name (since a device may have multiple accounts), and owner name.
- *Allowed Attendee Type:* A value such as *Optional* or *Required*.
- *Color, Location, Time Zone:* The calendar's color, location, and time zone (usually given as `+/-HH:MM`).
- *Is Primary:* Typically each device has one primary calendar and one or more secondary calendars. This value indicates whether the calendar is a primary calendar (**`true`**) or not (**`false`**).
- *Is Visible:* Whether the calendar is set to be visible (**`true`**) or not (**`false`**).
- *Sync Events:* Whether the calendar is set to sync down events (**`true`**) or not (**`false`**).

**Note:** If the calendar contains no information for a particular field, then nothing is returned for that field.

## Simulating the device's calendar

There are two options for simulating the device's calendar app:

- Microsoft Outlook's calendar
- An XML file that has the structure of the `$MT_CALENDAR` tree

Select the option you want to use in the Simulation tab of the Options dialog[1663] (**Tools | Options**).

⊟ Structure of a sample calendar file to use for simulations

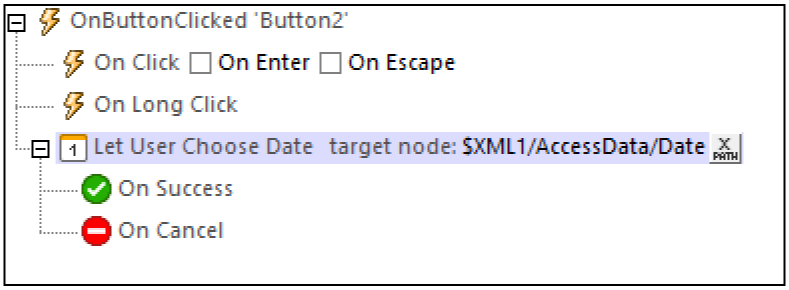
```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
   <Calendar Id="1" Name="Business">
      <Event Id="1" Title="Quarterly Meeting" Start="2018-04-04" End="2018-04-04"
AllDay="true()" Location="Meeting Room 2">
         <Attendee Name="Bob" Status="Accepted" Type="Required"
Relationship="Speaker"/>
      </Event>
      <Event Id="2" Title="New Customer Lunch" Start="2018-05-14T12:30:00" End="2018-
05-14T14:00:00" Location="Sushi Restaurant">
         <Attendee Name="Alice" Status="Accepted" Type="Optional"
Relationship="Attendee"/>
      </Event>
   </Calendar>
   <Calendar Id="2" Name="Private">
      <Event Id="1" Title="Family Dinner" Start="2018-05-18T19:00:00" End="2018-05-
18T23:00:00" Location="Home"/>
      <Event Id="2" Title="Summer Vacation" Start="2018-07-09" End="2018-07-22"
AllDay="true" Location="Home"/>
   </Calendar>
</Root>
```

### MobileTogether extension functions

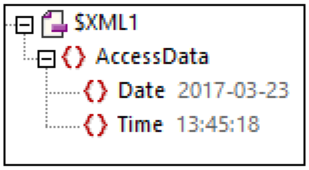
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.1.2 Let User Choose Date

The Let User Choose Date action (*see screenshot below*) causes a date-picker to be displayed on the client device. The date that the end user selects in the date-picker will be saved to the target page-source node that is specified in the action (*see screenshot*). The date will be saved in the format `YYYY-MM-DD`.


OnButtonClicked 'Button2'
On Click □ On Enter □ On Escape
On Long Click
Let User Choose Date target node: $XML1/AccessData/Date
On Success
On Cancel

The screenshot below shows the relevant page-source node in a simulation[1355], after a date has been selected on the client device.



### MobileTogether extension functions

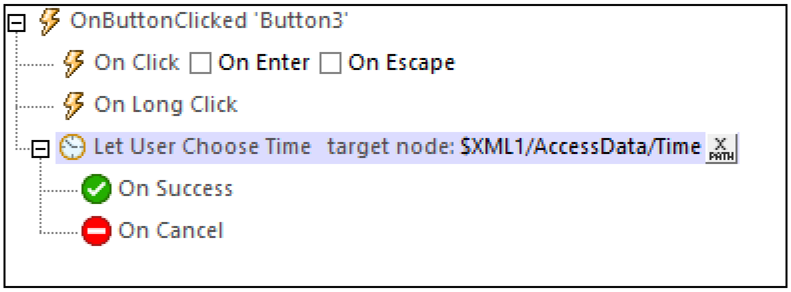
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-`**

**available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

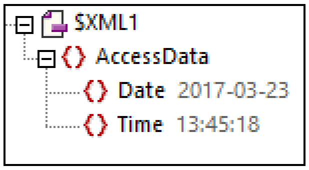
**mt-available-languages()**

## 10.1.3 Let User Choose Time

The Let User Choose Time action (*see screenshot below*) causes a clock to be displayed on the client device. The time that the end user selects in this clock will be saved to the target page-source node that is specified in the action (*see screenshot*). The time will be saved in the 24-hour format: `HH:MM:SS`.


OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
Let User Choose Time target node: $XML1/AccessData/Time
On Success
On Cancel

The screenshot below shows the relevant page-source node in a simulation[1355], after a time has been selected on the client device.

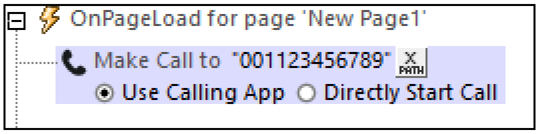


### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.1.4 Make Call To

Makes a call to the number specified in the XPath expression of the definition (*see screenshot below*).



OnPageLoad for page 'New Page1'

Make Call to "001123456789"

● Use Calling App ○ Directly Start Call

Note the following points:

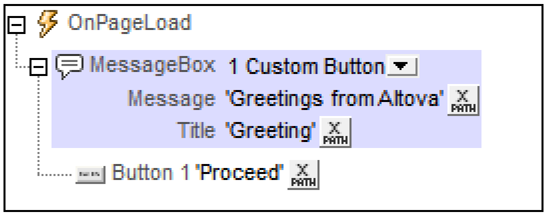
- The recipient's number is specified using an XPath expression.
- The recipient's number must be entered as an XPath string containing numbers and no spaces. A plus symbol at the beginning of a number is allowed. Example: `"004311234567"`.
- If the design is generated as an AppStore App[1471], then the following is possible: (i) the call is started directly *(Directly Start Call),* or (ii) the number is displayed in the device's telephone app, and the end user is asked whether to dial the number *(Use Calling App)*.
- If the design is deployed as a MobileTogether solution, then, on the action being triggered, the number is displayed in the device's telephone app and the end user is asked whether to dial the number. This happens even if the *Directly Start Call* option is selected.
- A static global variable named **`$MT_TelephonyAvailable`**[1300] can be used to test whether telephony services are available on the client device. Values of the variable can be **`true()`** or **`false()`**.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.1.5 Message Box

Defines a message box that is displayed when the event is triggered. The combo box enables you to select the buttons that appear in the message box. Predefined buttons are `OK`, `Yes`, `No`, `Cancel`. You can also define from one to three custom buttons. The text of custom buttons is specified in an XPath expression (for example, in the screenshot below left it is: **`"Proceed"`**).



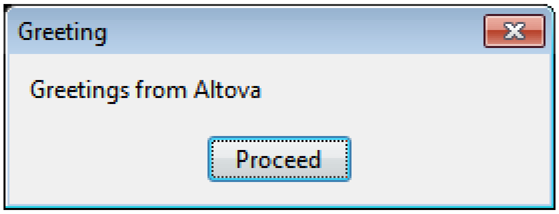
OnPageLoad

MessageBox 1 Custom Button

Message 'Greetings from Altova'

Title 'Greeting'

Button 1 'Proceed'



Greeting

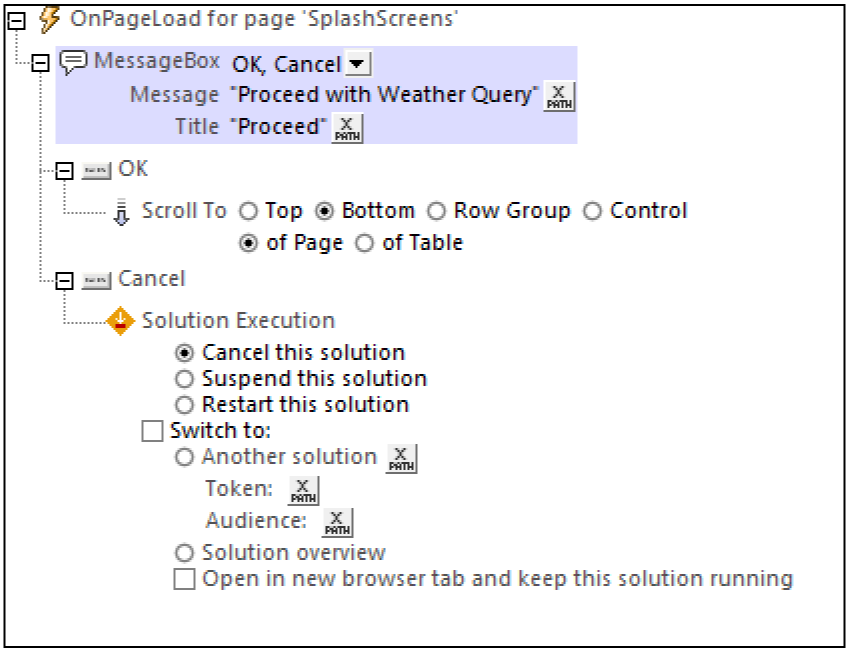
Greetings from Altova

Proceed

The screenshots above show a message box (*right*) and its definition (*left*). The definition is for a message box with a custom button. Notice how the title, message, and button text are defined.

## Message boxes with multiple user options

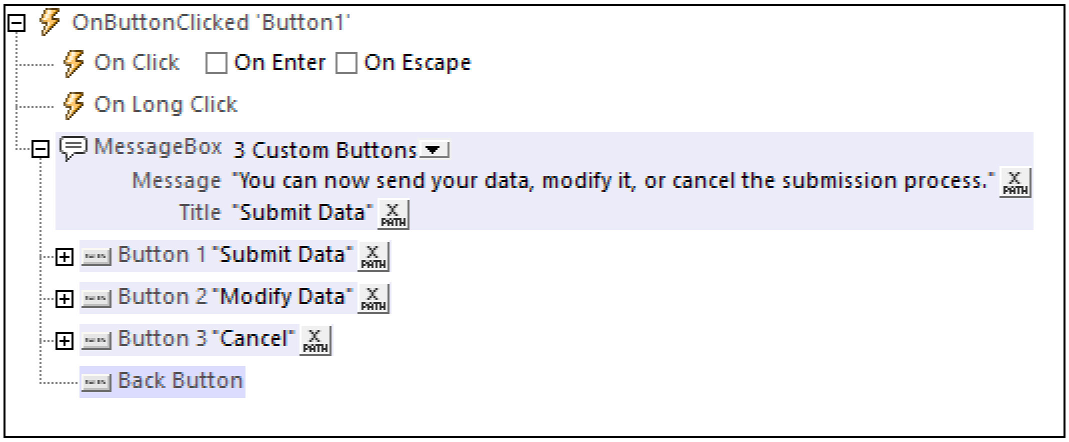
A message box can offer the user multiple options. This is done in the form of multiple buttons, each of which is linked to its own action. For example, in the screenshot below, two buttons, `OK` and `Cancel`, are defined by selecting them in the combo box. The buttons are automatically displayed in a hierarchical tree as child objects of MessageBox. Each button can then have actions assigned to them (*see screenshot*). These actions are carried out when the button is pressed by the user. For example, if the `Cancel` button defined in the screenshot below is clicked, then the End Solution action is carried out.



OnPageLoad for page 'SplashScreens'
MessageBox OK, Cancel
Message "Proceed with Weather Query"
Title "Proceed"
OK
Scroll To ○ Top ● Bottom ○ Row Group ○ Control
● of Page ○ of Table
Cancel
Solution Execution
● Cancel this solution
○ Suspend this solution
○ Restart this solution
□ Switch to:
○ Another solution
Token:
Audience:
○ Solution overview
□ Open in new browser tab and keep this solution running

## Custom buttons

Custom buttons provide you with the flexibility to configure a message according to the situation. You can create messages with one to three buttons. The example message below (*see screenshot*) uses three buttons, each of which is associated with a specific action. Additionally, you can also assign actions to the **Back** button of the mobile device. Do this by dragging the relevant actions under the **Back** button. If no action is assigned, then nothing will happen when the device's **Back** button is tapped.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
MessageBox 3 Custom Buttons
Message "You can now send your data, modify it, or cancel the submission process."
Title "Submit Data"
Button 1 "Submit Data"
Button 2 "Modify Data"
Button 3 "Cancel"
Back Button

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-available-languages()`**

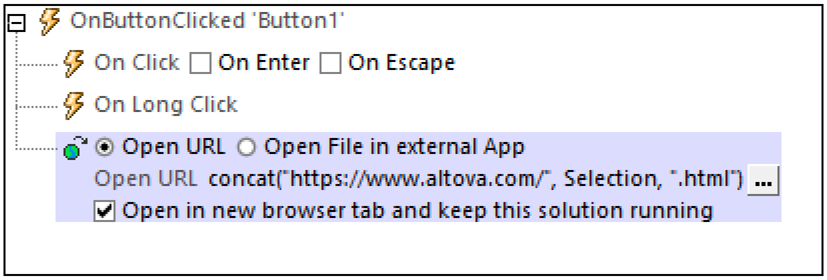
## 10.1.6 Open URL/File

Opens, in the client device, the URL or file that is specified in the action. A URL is opened in the client's default Internet browser. A file on the client device is opened in the client's default app for that file type. In the action's configuration options, select the *Open URL* or *Open File in external app* radio button and then enter details of the URL or file to open.

In both cases, an option is available for web clients to open the URL in the current tab (or, alternatively, in a new tab of the browser); *see screenshots below.*

### Open URL

In the XPath expression dialog of the *Open URL* action, if you select *Open URL*, then you can enter an expression that either: (i) evaluates to a string that is the URL to be opened (*first screenshot below*), (ii) evaluates to a string that constructs a data url (*second screenshot below*), or (iii) evaluates to a sequence of strings that constructs a command line instruction (for use in simulations in MobileTogether Designer only).



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
● Open URL ○ Open File in external App
Open URL concat("https://www.altova.com/", Selection, ".html") ...
☑ Open in new browser tab and keep this solution running

*Data URLs*

Data URLs enable you to open binary files directly in a new tab of the web client's browser. Note that these URLs can be accessed **in web clients only**. The syntax of a data URL is as follows:

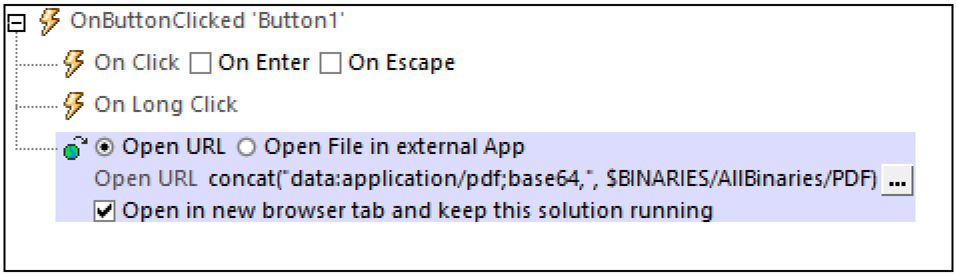
```
data:[<mediatype>][;base64],<data>
```

A data URL consists of four parts (colored differently above). The `data:` part is a fixed prefix. The `<mediatype>` part is a `MIME` type. The `;base64` token is optional; if used, it indicates that the `<data>` part must be read as Base64-encoded data; if omitted, the `<data>` part will be read as text. The Base64-encoded data can be read dynamically from a node in a page source (for an example, see below).

*Example*:

```
data:application/pdf;base64,<SomeBinaryData-SuchAsInThe-$BINARIES/PDF-Node>
```

In an XPath expression, a data URL that reads Base64-encoded data from a node can be entered as shown in the screenshot below. In this expression, the Base64-encoded data is contained in the **`$BINARIES/PDF`** node.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
● Open URL ○ Open File in external App
Open URL concat("data:application/pdf;base64,", $BINARIES/AllBinaries/PDF) ...
☑ Open in new browser tab and keep this solution running

**Note:** Base64-encoded data in a node can also be saved to file with the *Save to Binary File* 815 action, and the file can then be opened. The *Open Data URL* action, on the other hand, opens the file with one click only.

*Sequence of strings for a command line instruction*

The XPath expression for *Open URL* can also be constructed so that it evaluates to a command line instruction. In this case, the XPath expression must evaluate to a sequence of strings, where the first string would be the filepath to the executable. The parameters of the command line instruction can then be specified in: (i) a second string of the sequence, where each space-separated token is a command line parameter; (ii) subsequent strings of the sequence, where every subsequent string corresponds to one command line parameter; or (iii) a combination of the previous two patterns (that is, where there is more than one subsequent string, with each of these corresponding to one or more command line parameters).

**Note:** The parameters would have to be entered in the correct order.

**Note:** The option for constructing a command line instruction can only be used in a simulation in MobileTogether Designer. It will have no effect when run on a client.

*Examples:*

The two examples below call Altova DiffDog in order to compare two XML files. The first XPath expression is a sequence of two strings. The second is a sequence of three strings. In both cases, the command line instruction that is constructed will be the same.

```
("C:\Program Files\Altova\DiffDog2021\DiffDog.exe",
 "C:\TestFiles\Colors2.xml C:\TestFiles\Colors3.xml")

("C:\Program Files\Altova\DiffDog2021\DiffDog.exe",
 "C:\TestFiles\Colors2.xml",
 "C:\TestFiles\Colors3.xml")
```

**Note:** If a filepath contains a space, the filepath should be enclosed in quotes (single or double, as required).

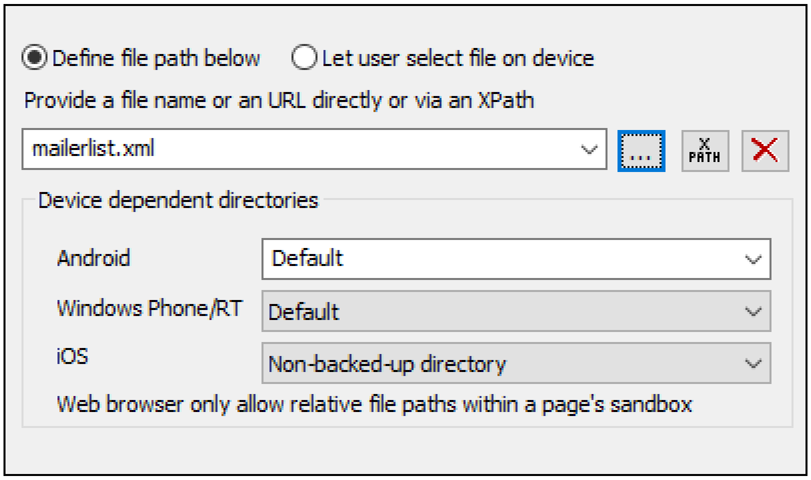
## Open file in external app

This action opens a file that is located on the client device. You can specify the file directly in the design, or you can let the end-user select the file. The file will be opened in the device's default app for the selected file's filetype. If the client is a web client, the file will be opened in a browser tab; an option is available to select either the current tab or a new tab. Note that, in web clients, file selection by the end-user is not available.

When you click the *Open File* action's **Edit** button, an Open dialog appears (*screenshot below*). Select the options that apply.

*File location is stored in design*

To directly specify (in the design) the device file to open, select *Define file path below* (*see next screenshot)*. Enter an absolute or relative file path, or an XPath expression that evaluates to such a file path. If you enter a relative path, then this path is resolved relative to the base directory you specify for that device type (*see list and screenshot below*).



● Define file path below ○ Let user select file on device

Provide a file name or an URL directly or via an XPath

mailerlist.xml

Device dependent directories

Android: Default

Windows Phone/RT: Default

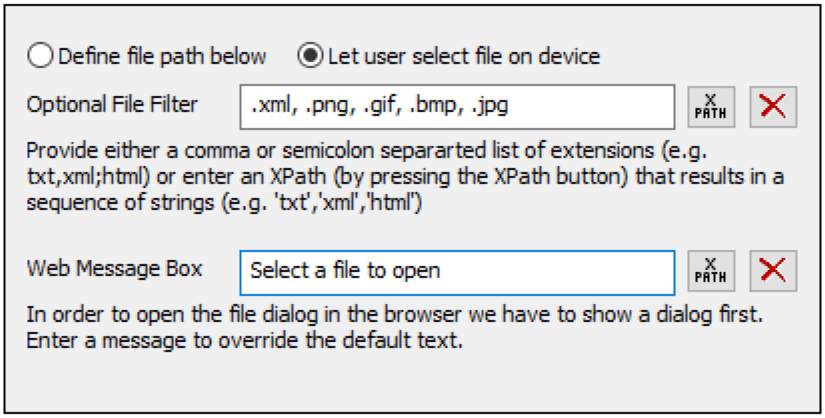
iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

- *Android:* Select the Android device-directory from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. Note, however, that, unless the Android device is rooted, no other app (besides MobileTogether) will be able to access the MobileTogether sandbox directory. So, trying to open a file in the MobileTogether sandbox with another app could fail.
- *Windows RT:* Select the Windows Phone or Windows RT device folder from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected.
- *iOS:* MobileTogether creates two directories on the user's device: (i) a *Backed-up directory* for files that are backed up by the operating system and will be available at later times, for example, after a restore. This directory is intended for files that are important for the user and that should not be lost; (ii) a *Non-backed-up directory* for files that do not need to be backed up, or if faster buffering is needed for playback. Select *Backed-up directory* or *Non-backed-up directory* as required.
- *Web browser:* No selection is available. Relative paths are resolved within the context of the browser's sandbox.

*User selects device file*

To let the end user select a file on the mobile device, select *Let user select file on device* (*see screenshot below*). When the action is processed at runtime, the end user can browse for, or enter the name of, a device file to open.



○ Define file path below ● Let user select file on device

Optional File Filter: .xml, .png, .gif, .bmp, .jpg

Provide either a comma or semicolon separarted list of extensions (e.g. txt,xml;html) or enter an XPath (by pressing the XPath button) that results in a sequence of strings (e.g. 'txt','xml','html')

Web Message Box: Select a file to open

In order to open the file dialog in the browser we have to show a dialog first. Enter a message to override the default text.

This option provides the following options:

- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be opened so that only those file extensions that you have defined here are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).
- *Web Message Box:* Before the File Open dialog is opened on the client device, a message box is displayed. You can enter a message to override the default text of this message box. Enter the text of the message directly, or via an XPath expression.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable [907]. The Catch part of the Try/Catch action [907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action [907] for details.

An error is registered only if the file does not exist. If the file exists, a success is registered—even if the file could not be opened by any device app.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions [1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

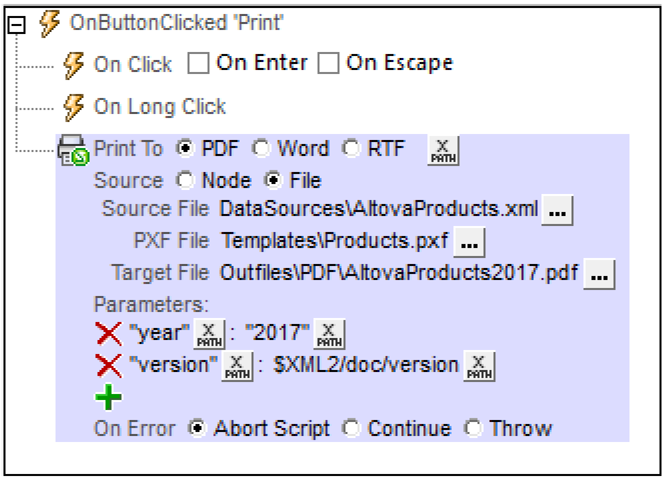
**`mt-extract-file-extension()`**
**`mt-extract-file-name()`**

## 10.1.7 Print To

The Print To action (*screenshot below*) uses Altova StyleVision Server (version 2017sp1 or later) to generate a PDF, Word, or RTF output document. The mechanism works as follows: XML data in one of the design's page source nodes or in an external XML file is processed with a PXF file, which is essentially an XSLT stylesheet. Altova StyleVision Server is used to carry out this XSLT transformation. In order for the Print To action to be carried out, the StyleVision Server application, which is available for download at the Altova website, must be installed on the same computer as MobileTogether Designer (for local simulation) and as MobileTogether Server (for server simulations and deployed use).

*Note about PXF files and StyleVision Server*

- A Portable XML Form (PXF) file is a file format that has been specially developed by Altova to package XSLT stylesheets for multiple outputs into one file. The various XSLT stylesheets are generated from a single design created in Altova StyleVision.
- In the design, you can define parameters, which are then also defined in the XSLT stylesheets contained in the PXF file. At runtime, values can be passed to these parameters in the stylesheets.
- StyleVision Server is a lightweight command-line application that is used to run XSLT transformations that generate output in various formats.
- For more information about the PXF format, see the Altova StyleVision documentation. For more information about StyleVision Server, see the Altova StyleVision Server documentation.



⊟ OnButtonClicked 'Print'
On Click □ On Enter □ On Escape
On Long Click
Print To ● PDF ○ Word ○ RTF
Source ○ Node ● File
Source File DataSources\AltovaProducts.xml ...
PXF File Templates\Products.pxf ...
Target File Outfiles\PDF\AltovaProducts2017.pdf ...
Parameters:
"year" : "2017"
"version" : $XML2/doc/version
On Error ● Abort Script ○ Continue ○ Throw

The Print To action takes the following options:

*PDF/Word/RTF*
Specifies the type of print output document: `.pdf`, `.docx`, or `.rtf`. You can also use an XPath expression to select the output. The result of the evaluation should be one of the following strings: **`"PDF"`**, **`"Word"`**, or

**"RTF"** (case-insensitive). Using an XPath evaluation enables you to make the selection of print format conditional.

*Source*
You can select either: (i) an XML node of one of the design's page sources, or (ii) an external XML file, which can be on the client device or on the server.

*Source Node*
If you choose *Node* as your source, then enter an XPath expression to select the node you want to use as source data for the output document. The node you select will be entered as the value of this option, and can be changed subsequently.

*Source File*
If you choose *File* as your source, the path to the file you select will be entered as the value of the Source File option. You can also add a filepath (or modify an already entered filepath) by clicking the **Edit** button of the *Source File* option. A dialog appears in which you can select a server file or client file. See the section *File locations* 688 below for details of the available options. On selecting a file that is defined to reside on the server, you are asked whether the file should be deployed to the server or not 289. If the file is not deployed, it must be stored in the solution's working directory 289 (or a descendant directory); in this case, the path that you enter for this (*Source File*) option must be correctly set 289 so that the solution correctly accesses the source file at runtime.

*PXF File*
The PXF file is the stylesheet container file that StyleVision Server uses to generate the output document. Select the PXF file using one of the methods described for server locations in the *File locations* 688 section below. If you do not deploy the PXF file, make sure that you save it in the solution's working directory (or a descendant directory); in this case, the path that you enter for this (*Source File*) option must be correctly set 289 so that it correctly accesses the PXF file at runtime..

*Target File*
Specifies the name of the output file and its location (on server or client). Use any one of the ways described in the section File locations 688 below to specify the file's location.

*Parameters*
At runtime, parameter values can be passed to the stylesheets in the PXF file. This option enables you to specify multiple parameter values. Click the **Add Parameter** icon to add a parameter entry. Then enter the parameter's name and value as XPath expressions. In the screenshot above, for example, the first parameter has a **name:value** pair of **"year":"2017"**. The second parameter takes its value from the node `$XML2/doc/version`. You can add as many parameter values as you like.

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

## File locations

When you click the **Additional Dialog** button of the *Source File*, *PXF File*, and *Target File* options, a Specify File dialog appears in which you can specify the file to load or save, respectively.

- *Source file:* Selects an XML data file that is located on the server or client. If you select a file on a local or network machine, make sure that you either: (i) deploy the file with the design to the server, or (ii) store the file to the solution's working directory on the server (or a descendant directory).
- *PXF file:* Selects a PXF file that is located on the server. If you select a file on a local or network machine, make sure that you either: (i) deploy the file with the design to the server, or (ii) store the file to the solution's working directory on the server (or a descendant directory). A client-based PXF file cannot be specified.
- *Target file:* Generates the output to a server or client location.

The available options in the Specify File dialog depends on whether the file is being loaded (*Source File* and *PXF File*) or saved (*Target File*).

*File is located on server*

The term server location refers to the location of the file at runtime. When you select a file to load, you can select it from any location on your network. However, in order to make the file accessible at runtime, you must either: (i) deploy the file to the server, or (ii) save it to the solution's working directory on the server. To specify the file to load, you can either browse for its location (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). In the dialog, select the options you want.

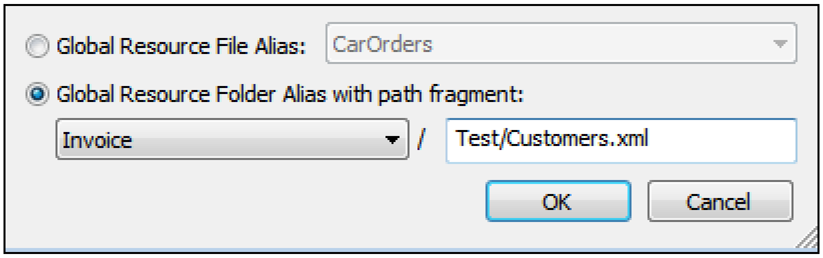
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). See the section Altova Global Resources 1334 for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

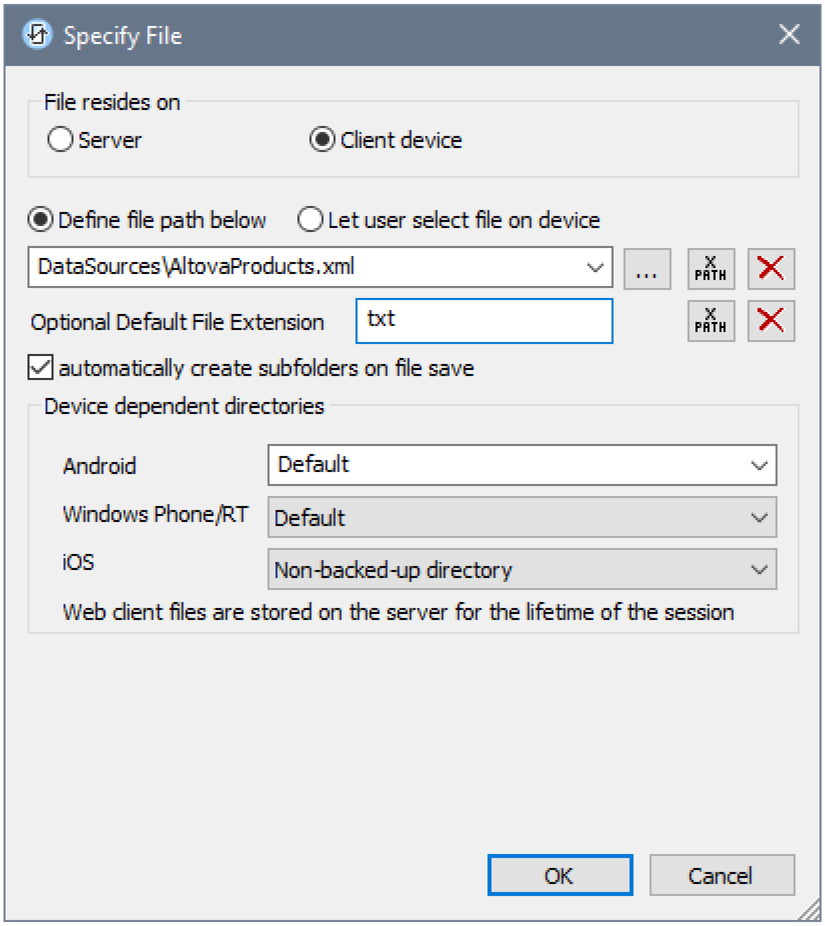
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 1334 for details.

*File is located on client*

If the file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



Specify File

File resides on

○ Server ● Client device

● Define file path below ○ Let user select file on device

DataSources\AltovaProducts.xml

Optional Default File Extension: txt

☑ automatically create subfolders on file save

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web client files are stored on the server for the lifetime of the session

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To[686] (*Source File* and *Target File* options), Load/Save File[809], Load/Save Image[707], Load/Save Binary File[815], Load/Save Text File[821], Read Folder[847], and Get File Info[849].

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.
- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the

directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog[1663] (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).

- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

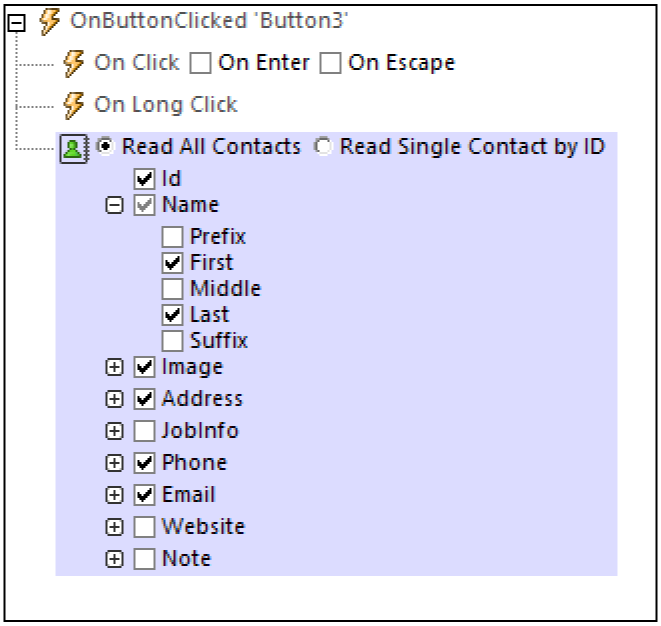
  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.1.8 Read Contacts

When a Read Contacts action (*screenshot below*) is added to the design, the `$MT_CONTACTS` page source tree is automatically added to the design. When the action is triggered, contacts from the device's address book are read and stored in the `$MT_CONTACTS` tree.



OnButtonClicked 'Button3'

On Click ☐ On Enter ☐ On Escape

On Long Click

● Read All Contacts ○ Read Single Contact by ID

- ☑ Id
- ☑ Name
  - ☐ Prefix
  - ☑ First
  - ☐ Middle
  - ☑ Last
  - ☐ Suffix
- ☑ Image
- ☑ Address
- ☐ JobInfo
- ☑ Phone
- ☑ Email
- ☐ Website
- ☐ Note

When defining the action, you can specify the following:

- Whether to read all contacts, or only one contact on the basis of its ID
- What fields from each contact's data to read and store. Do this by checking the fields that should be read (*see screenshot*).

**Note:** IDs are platform-dependent (and might even be different among versions of a single platform). So, in order to find the ID of a given contact, you will need to read out all contacts (with their IDs) and locate the required ID on the basis of other fields.

**Note:** Reading all the fields of all the contacts can consume considerable time and memory. It is recommended therefore to narrow the read-out list to only the fields and contacts that are needed. You can do this, for example, in a two-step process: (i) read names and their IDs; (ii) for the final read-out, read only the desired fields of the desired IDs.

**Note:** To simulate a device's address book (in order to run simulations), you can create and use a sample contacts file (see Contacts Sample Files[1381] for details). Alternatively, you can use the contacts of your Microsoft Outlook application by selecting the corresponding option in the Simulation tab of the Options dialog[1663].

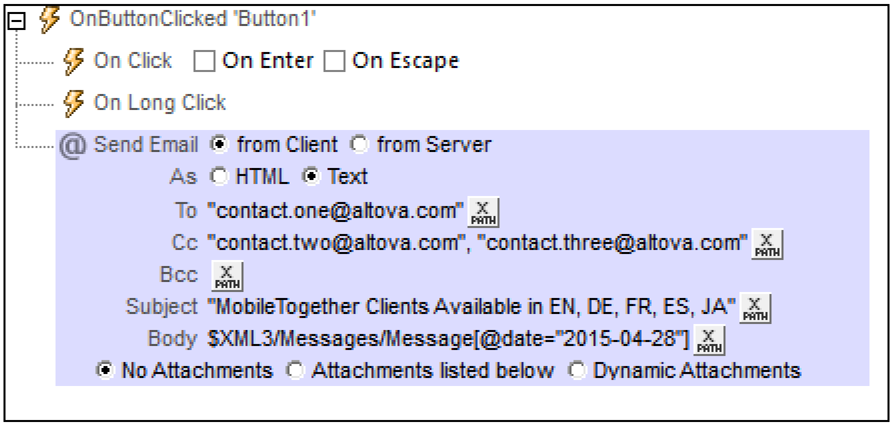
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-`**

**available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.1.9 Send Email To

Sends an email from the email application on the mobile device or silently via the server to one or more recipients. The email can be sent as HTML or text. You can specify the recipients, the subject, and the message of the email body via XPath expressions. Additionally, text and image attachments can be generated. The settings of the Send Email To action are shown in the screenshot below and are described further below.

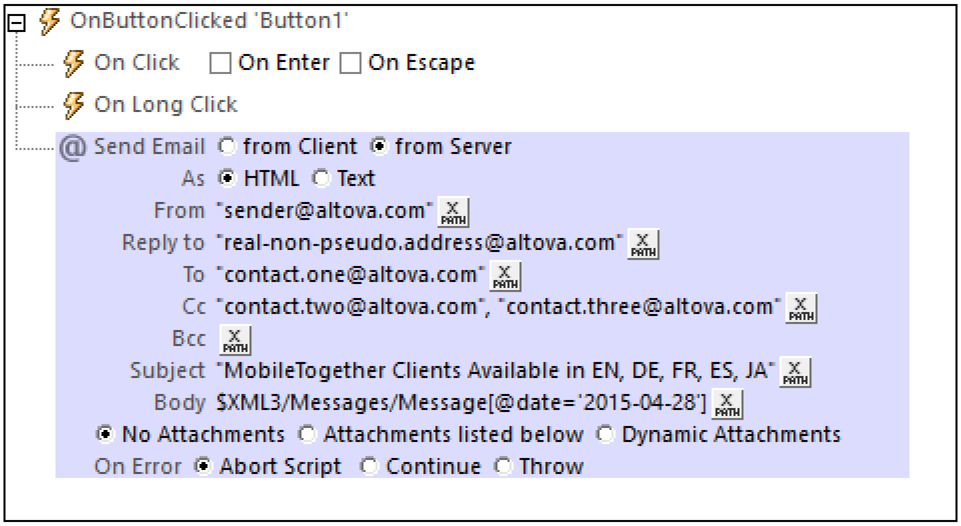


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Send Email ● from Client ○ from Server
As ○ HTML ● Text
To "contact.one@altova.com"
Cc "contact.two@altova.com", "contact.three@altova.com"
Bcc
Subject "MobileTogether Clients Available in EN, DE, FR, ES, JA"
Body $XML3/Messages/Message[@date="2015-04-28"]
● No Attachments ○ Attachments listed below ○ Dynamic Attachments

**Note:** Links to update server settings do not work in Gmail and some other email applications, but they work in popular clients such as AquaMail, K9, and MailWise. They have been tested in AquaMail and K9 and work correctly in these applications. The `mobiletogether://` scheme is used for MobileTogether-specific tasks such as opening a MobileTogether solution via the link[1239] or updating the server settings on a client device via the link[291].

▼ Send email from client or server

Select whether the email is to be sent from the email application of the end user's client device or from the server. Compare the screenshot below (email sent from server) with the screenshot at the top of the page (client); the server option has three settings more than the client option: the *From* field, the *Reply to* field, and the *OnError* action.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Send Email ○ from Client ● from Server
As ● HTML ○ Text
From "sender@altova.com"
Reply to "real-non-pseudo.address@altova.com"
To "contact.one@altova.com"
Cc "contact.two@altova.com", "contact.three@altova.com"
Bcc
Subject "MobileTogether Clients Available in EN, DE, FR, ES, JA"
Body $XML3/Messages/Message[@date='2015-04-28']
● No Attachments ○ Attachments listed below ○ Dynamic Attachments
On Error ● Abort Script ○ Continue ○ Throw

- *Client:* When the action is executed, it opens an email in the email application. The email will be filled with the details specified in the action settings (address fields, subject, body, and attachments). The end user can edit the email and send it, or close the email without sending it. You can also specify whether the email is sent as HTML or text. Note that for Windows Phone clients there is a limit of 2083 characters (for subject + addresses + body); any characters beyond this limit will not be received.
- *Server:* If the email is to be sent via the (MobileTogether) server, then MobileTogether Server must be configured to access the ISP's SMTP server. (See the MobileTogether Server documentation for a description of how to do this. Essentially, the ISP's SMTP server address and port, and the sender's email user name and password must be configured in the settings of MobileTogether Server.) When the end user carries out the event that triggers the action, the email is sent silently from the server without any further end-user interaction. Choosing to send the email via the server provides three options more than for sending from the client: the *As* field, the *From* field, and the *On Error* action to perform. The *As* field specifies whether the email should be sent as HTML or text. The *From* field is described below. If there is an error when sending the mail from the server, you can specify whether the Send action should be aborted, continued, or throw an error.

### To, Cc, Bcc

The email addresses that go into these fields are entered via XPath expressions. They can be (i) entered directly as strings in the XPath expression (as shown in the screenshots above), or (ii) generated from nodes in page sources (*see the XPath expression below*). If multiple recipients are to be specified in any of these fields, then it is best to use an XPath expression that returns a sequence. It is not advisable to hard-code separators (such as semi-colons or commas) between two email addresses since different email clients use different separators. Here is an example of an XPath expression that uses a node in an XML page source to generate an email address line:

```
if ( $MT_iOS=true() ) then iosGroup/Person/Email else otherGroup/Person/Email
```

The expression iterates over a sequence of `Person/Email` nodes, each of which is expected to contain one email address. In the case of both iOS clients and non-iOS clients, multiple recipients are given as a sequence of strings: `("contact1@altova.com", "contact2@altova.com")`.

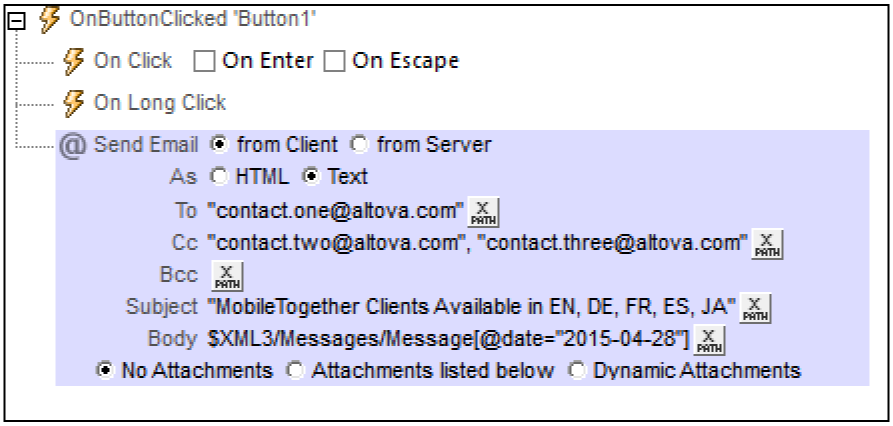
### From and Reply To

If you choose to send the email via MobileTogether Server, then the *From* and *Reply To* settings become available. You can specify the sender's email address in this setting. The *From* setting must be filled if the email is sent over an SMTP server that requires a sender's address as mandatory. If the SMTP server does not require this, you can leave the *From* setting empty.

Automatic replies are often sent from a "pseudo" email address, which is the address made visible to recipients. If you wish to use a "pseudo" address, enter this address  in the *From* setting. If, in this case, you still wish to be contactable, you can enter a real email address in the *Reply To* setting. When the recipient clicks the **Reply** command in their email client, a new email will open that is addressed to the real email address that was entered in the *Reply To* setting.

### Email subject field and body

The XPath expression for the email's *Subject* field can either be a string or generate the desired text from XML page sources. The XPath expression for the email's *Body* field must generate structured HTML, that is, an `<html>` element with a valid HTML sub-structure (*see below*).



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Send Email ● from Client ○ from Server
As ○ HTML ● Text
To "contact.one@altova.com"
Cc "contact.two@altova.com", "contact.three@altova.com"
Bcc
Subject "MobileTogether Clients Available in EN, DE, FR, ES, JA"
Body $XML3/Messages/Message[@date="2015-04-28"]
● No Attachments ○ Attachments listed below ○ Dynamic Attachments

In the screenshot above, the XPath expression for the *Subject* field is a directly entered string, while the expression for the body returns the content of the `Message` element that has its `date` attribute equal to `"2015-04-15"`.

The email body must be structured HTML, that is, it must be structured as an `<html>` element that contains child nodes. The level of HTML support depends on the source from which the email is sent:

- *From the server or from an iOS client:* Standard HTML supported.
- *Android:* Only a few HTML constructs, such as `<b>`, are supported.
- *Windows Phone and Windows App (tablets and touch-enabled PCs)*: No HTML support.

If you wish to send fully formatted HTML, select the (*Send) from Server* option.

### Attachments

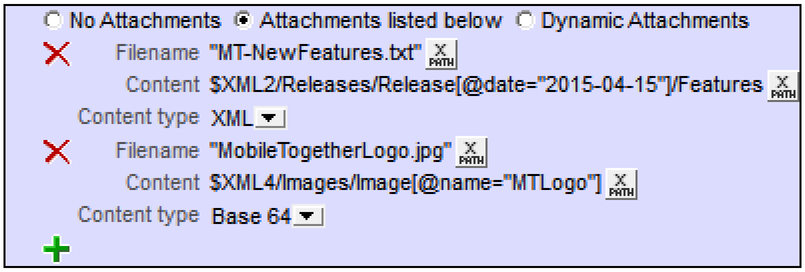
Files and images can be attached to the message. You can select one of three options for attachments:

- No attachments (selected by default)
- Attachments listed below
- Dynamic attachments

**Note:** Note the following points for Windows Phone and Windows App clients: Windows Phone supports attachments; Windows 8.1 does not. On Windows 10, the default Outlook Mail client that is installed with the OS supports attachments, but if Microsoft Outlook is the default mail client, then attachments are not supported.

*Attachments listed below*

This option allows attachments to be created individually. To add a new attachment, click +. The screenshot below shows a message with two attachments. To delete an attachment, click its **Delete** icon.



○ No Attachments ● Attachments listed below ○ Dynamic Attachments

✕ Filename "MT-NewFeatures.txt"
Content $XML2/Releases/Release[@date="2015-04-15"]/Features
Content type XML

✕ Filename "MobileTogetherLogo.jpg"
Content $XML4/Images/Image[@name="MTLogo"]
Content type Base 64

+

Each attachment has the following properties:

- *Filename (XPath):* The filename can have any extension. The filename serves solely as a representation (in the message) of the attachment; it is not an actual path.
- *Content (XPath):* You can select an XML tree fragment, a single XML node, the text content of one or more nodes, or you can directly enter a string that will be the content of the attached file. The content will be parsed according to the selection in the (next) *Content type* property.
- *Content type (combo box: XML/Base64/Text):* If the content type is `XML`, then the content that is selected via the Content property (*previous property*) is parsed as XML data: an XML nodeset is expected, and the nodeset will be attached to the email. If the content type is `Base64`, then Base64-encoded content is expected, and this content is decoded. So, if the content is a Base64-encoding of an image, then an image is generated and attached to the email. If the content type is `Text`, then text is expected as content, and this text will be attached to the email. Note that the value of the *Content* property must be readable according to the selection made for the *Content type* property.

*Dynamic attachments*

The XPath expression uses the **`mt-email-attachment`**[1262] XPath extension function to create the attachments.

▼ mt-email-attachment

```
mt-email-attachment(Filename as xs:string, Content as item(), ContentType as
xs:string) as array(*)
```

Prepares the XML, Base64, or text content that is provided by the **`Content`** argument as an email attachment. Whether the content is parsed as XML or Base64 or text is determined by the

`ContentType` argument, which takes either `XML`, `Base64`, or `text` as its value. The filename that is associated with the attachment is given by the **`Filename`** argument.

**Note:** The `mt-email-attachment` is a requirement when using the *Dynamic Attachments* option of the Send Email To[693] and Share[699] actions.

**Note:** For emails that are sent as HTML, the email body must be correct HTML, that is, it must start with the `html` element. A valid body could be created for example with the following XPath/XQuery construct: `element html { element body { "Test" } }`

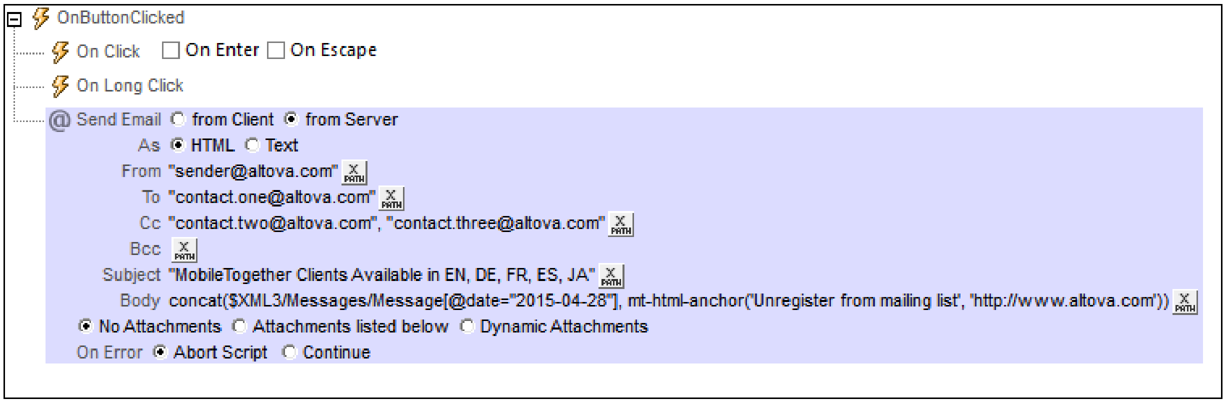
**Note:** Attachments work with Android and iOS clients only.

⊟ *Examples*

- `mt-email-attachment('MTNewFeatures.txt', $XML2/Releases/Release[@date='2015-04-15']/Features, 'XML')` returns `the Features node`
- `mt-email-attachment('MTLogo.jpg', $XML4/Images/Image[@name='MTLogo'], 'Base64')` returns `an image file`

### Adding links to the email body

You can add a hyperlink to the body of an email that is sent in HTML format. This feature does not work for emails sent as text. The link can target an Internet page or a MobileTogether solution. To add a link to the email body, use the **`mt-html-anchor`**[1262] function in the XPath expression of the *Body* option (*see screenshot below*).



OnButtonClicked
On Click □ On Enter □ On Escape
On Long Click
Send Email ○ from Client ● from Server
As ● HTML ○ Text
From "sender@altova.com"
To "contact.one@altova.com"
Cc "contact.two@altova.com", "contact.three@altova.com"
Bcc
Subject "MobileTogether Clients Available in EN, DE, FR, ES, JA"
Body concat($XML3/Messages/Message[@date="2015-04-28"], mt-html-anchor('Unregister from mailing list', 'http://www.altova.com'))
● No Attachments ○ Attachments listed below ○ Dynamic Attachments
On Error ● Abort Script ○ Continue

The **`mt-html-anchor`**[1262] function takes two arguments: `LinkText` and **`targetURL`**. It uses these two arguments to create an HTML hyperlink element: `<a href="targetURL">LinkText</a>`

For example:

```
mt-html-anchor('Unregister from mailing list', 'http://www.altova.com')
```

generates the HTML code fragment:

```
<a href="http://www.altova.com'">Unregister from mailing list</a>
```

The example above links to an Internet page. For a description of how to link to a MobileTogether solution, see Hyperlinking to Solutions[1239].

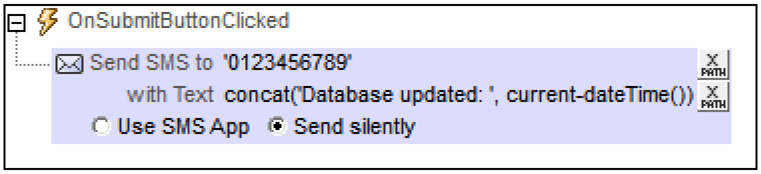
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-email-attachment()`**

# 10.1.10 Send SMS To

Sends an SMS containing the specified text to the specified telephone number *(see screenshot below)*.



OnSubmitButtonClicked

Send SMS to '0123456789'

with Text concat('Database updated: ', current-dateTime())

○ Use SMS App ● Send silently

Note the following points:

- The recipient's number and the SMS text are specified using XPath expressions.
- The recipient's number must be entered as an XPath string containing numbers. Example expression: `"004311234567"`.
- If multiple recipients are planned, the XPath expression must be a sequence of string items, for example: `("+4311234567", "0011123456789")`.
- If the design is generated as an AppStore App[1471], then you can choose whether to send the SMS silently *(Send silently)* or whether the end user sends the SMS, which will be opened in the device's SMS app and await the user's decision *(Use SMS App)*. Note that, for the *Send silently* option to work, (i) the AppStore App will need the *Send SMS* privilege, and (ii) the end user's permission will be asked about whether the app may send messages.
- If the design is deployed as a MobileTogether solution, then the message is opened in the device's SMS app, and the end user is asked whether the message should be sent. This happens even if the *Send silently* option is selected.
- A static global variable named **`$MT_SMSAvailable`**[1300] can be used to test whether SMS services are available on the client device. Values of the variable can be **`true()`** or **`false()`**.

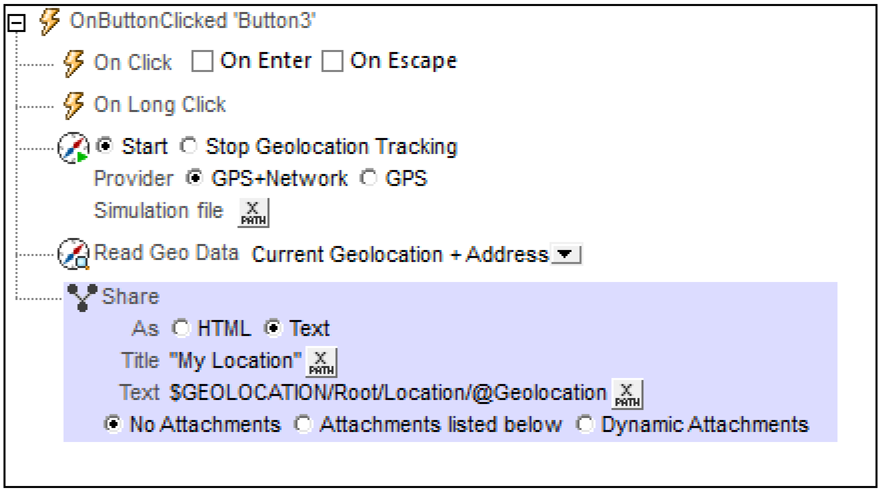
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a

full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.1.11 Share

The Share action (*highlighted in the screenshot below*) enables the end user to share text and images. The text can be sent as HTML or plain text, and it is selected via an XPath expression in the Share action's *Text* field (*see screenshot below*). Additionally, text and image attachments can be generated. When the Share action is triggered on the mobile device, the device's sharing options are displayed. (These are the messaging and social-networking apps that are installed on the mobile device.) The end user can then select one of these apps, and proceed as he or she usually would when sharing.



OnButtonClicked 'Button3'

On Click ☐ On Enter ☐ On Escape

On Long Click

● Start ○ Stop Geolocation Tracking

Provider ● GPS+Network ○ GPS

Simulation file

Read Geo Data Current Geolocation + Address

Share

As ○ HTML ● Text

Title "My Location"

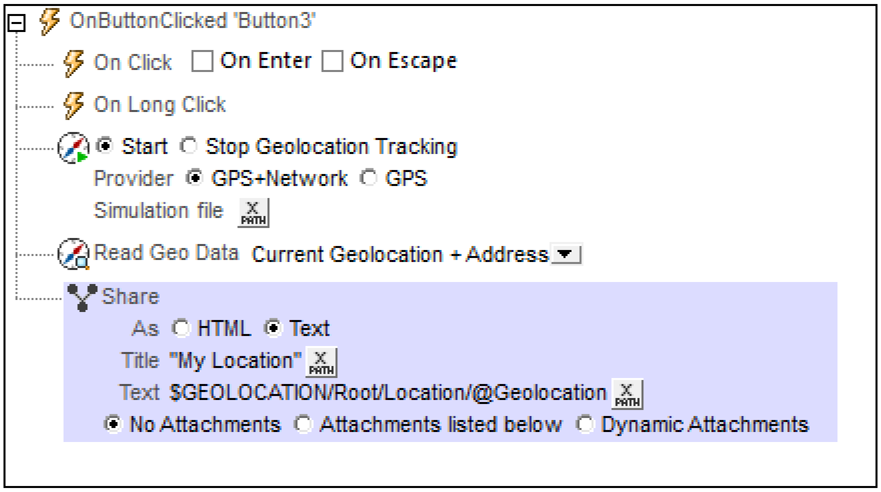
Text $GEOLOCATION/Root/Location/@Geolocation

● No Attachments ○ Attachments listed below ○ Dynamic Attachments

**Note:** The Share action cannot be used on web clients.

▼ Message title and text

The XPath expressions for these two settings (the message's title and body text) can either be a string or can generate the respective texts from XML page sources.


OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
● Start ○ Stop Geolocation Tracking
Provider ● GPS+Network ○ GPS
Simulation file
Read Geo Data Current Geolocation + Address
Share
As ○ HTML ● Text
Title "My Location"
Text $GEOLOCATION/Root/Location/@Geolocation
● No Attachments ○ Attachments listed below ○ Dynamic Attachments

In the screenshot above, the XPath expression for the *Title* field is a directly entered string, while the expression for the body text returns the content of the `Location/@Geolocation` node. This node provides the geolocation coordinates of the mobile device, which were obtained by using the Start/Stop Geo Tracking[735] and Read Geo Data[736] actions (*see screenshot above*).

**Note:** On iOS, the HTML/text selection has no effect; some apps might automatically interpret an existing `html` flag correctly.

Attachments

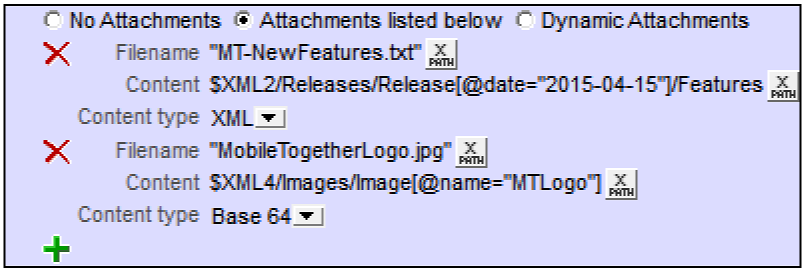
Files and images can be attached to the message. You can select one of three options for attachments:

- No attachments (selected by default)
- Attachments listed below
- Dynamic attachments

**Note:** Note the following points for Windows Phone and Windows App clients: Windows Phone supports attachments; Windows 8.1 does not. On Windows 10, the default Outlook Mail client that is installed with the OS supports attachments, but if Microsoft Outlook is the default mail client, then attachments are not supported.

*Attachments listed below*

This option allows attachments to be created individually. To add a new attachment, click ➕. The screenshot below shows a message with two attachments. To delete an attachment, click its **Delete** icon.



○ No Attachments ● Attachments listed below ○ Dynamic Attachments

Filename "MT-NewFeatures.txt"
Content $XML2/Releases/Release[@date="2015-04-15"]/Features
Content type XML

Filename "MobileTogetherLogo.jpg"
Content $XML4/Images/Image[@name="MTLogo"]
Content type Base 64

Each attachment has the following properties:

- *Filename (XPath):* The filename can have any extension. The filename serves solely as a representation (in the message) of the attachment; it is not an actual path.
- *Content (XPath):* You can select an XML tree fragment, a single XML node, the text content of one or more nodes, or you can directly enter a string that will be the content of the attached file. The content will be parsed according to the selection in the (next) *Content type* property.
- *Content type (combo box: XML/Base64/Text):* If the content type is `XML`, then the content that is selected via the Content property (*previous property*) is parsed as XML data: an XML nodeset is expected, and the nodeset will be attached to the email. If the content type is `Base64`, then Base64-encoded content is expected, and this content is decoded. So, if the content is a Base64-encoding of an image, then an image is generated and attached to the email. If the content type is `Text`, then text is expected as content, and this text will be attached to the email. Note that the value of the *Content* property must be readable according to the selection made for the *Content type* property.

*Dynamic attachments*
The XPath expression uses the **`mt-email-attachment`**[1262] XPath extension function to create the attachments.

▼ mt-email-attachment

**mt-email-attachment(Filename** *as xs:string*, **Content** *as item()*, **ContentType** *as xs:string*) **as array(*)**
Prepares the XML, Base64, or text content that is provided by the **`Content`** argument as an email attachment. Whether the content is parsed as XML or Base64 or text is determined by the **`ContentType`** argument, which takes either `XML`, `Base64`, or `text` as its value. The filename that is associated with the attachment is given by the **`Filename`** argument.

**Note:** The `mt-email-attachment` is a requirement when using the *Dynamic Attachments* option of the Send Email To[693] and Share[699] actions.

**Note:** For emails that are sent as HTML, the email body must be correct HTML, that is, it must start with the `html` element. A valid body could be created for example with the following XPath/XQuery construct: **`element html { element body { "Test" } }`**

**Note:** Attachments work with Android and iOS clients only.

⊟ *Examples*

- `mt-email-attachment('MTNewFeatures.txt', $XML2/Releases/Release[@date='2015-04-15']/Features, 'XML')` returns `the Features node`

- `mt-email-attachment('MTLogo.jpg', $XML4/Images/Image[@name='MTLogo'], 'Base64')` returns `an image file`

The tutorial Sharing Geolocations[230] shows how the Share action can be used.

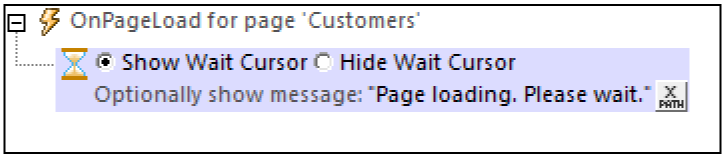
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-client-ip-address()`**

## 10.1.12 Wait Cursor

When the Show Wait-Cursor action (*see screenshot below*) is triggered, a platform-dependent wait cursor is displayed on the client. Optionally, an additional message can be displayed simultaneously. The display of the wait-cursor continues till the Hide Wait-Cursor action is triggered. If you think that a MobileTogether task might take long, then displaying the wait-cursor is useful for informing the user that a task is in progress.


OnPageLoad for page 'Customers'
● Show Wait Cursor ○ Hide Wait Cursor
Optionally show message: "Page loading. Please wait."

Use the wait-cursor action as follows:

1. Add the Show Wait-Cursor action (the Wait Cursor action with *Show Wait Cursor* selected) before the action for which you wish to use the wait-cursor.
2. Add the action or actions for which you wish to use the wait-cursor. Add as a sibling (or child) action of the Show Wait-Cursor action.
3. Add the Hide Wait-Cursor action (the Wait Cursor action with *Hide Wait Cursor* selected). (Note that the Wait Cursor will be hidden automatically once the actions for which it is displayed (Step 2) have all been completed.)

When the event containing this sequence of actions is triggered, the following happens: (1) the wait-cursor is displayed; (2) the actions for which you wish to use the wait-cursor are executed; (3) when these actions are completed, then the display of the Wait Cursor is stopped automatically; the Hide Wait-Cursor action can also be used to stop the cursor display, but is not a necessity.

## MobileTogether extension functions

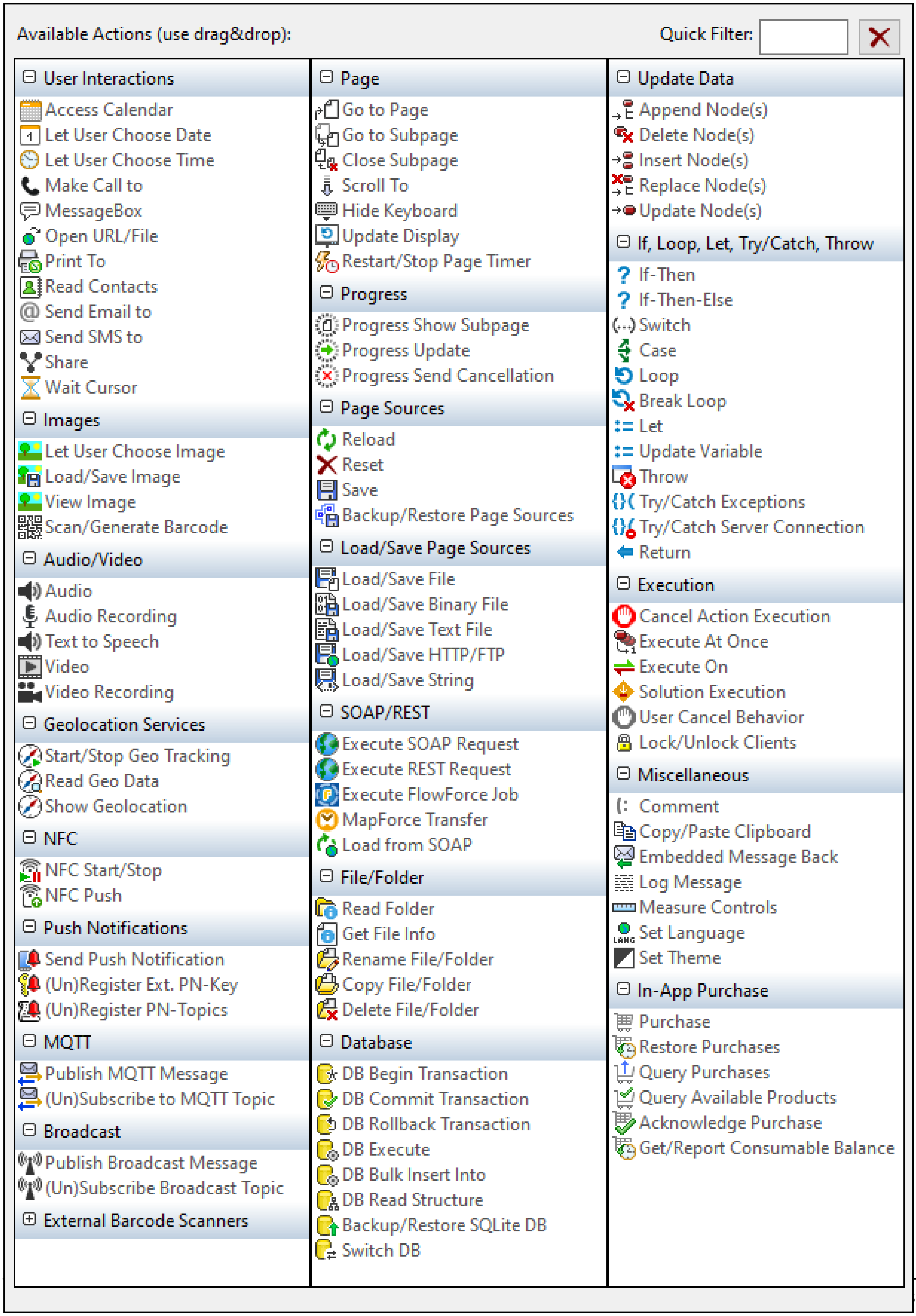
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-wait-cursor-shown()`**

# 10.2 Images

The following actions are available in the Images group of the Actions dialog (*screenshot below*):

- Let User Choose Image 706
- Load/Save Image 707
- View Image 713
- Scan/generate Barcode



Available Actions (use drag&drop): Quick Filter:

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

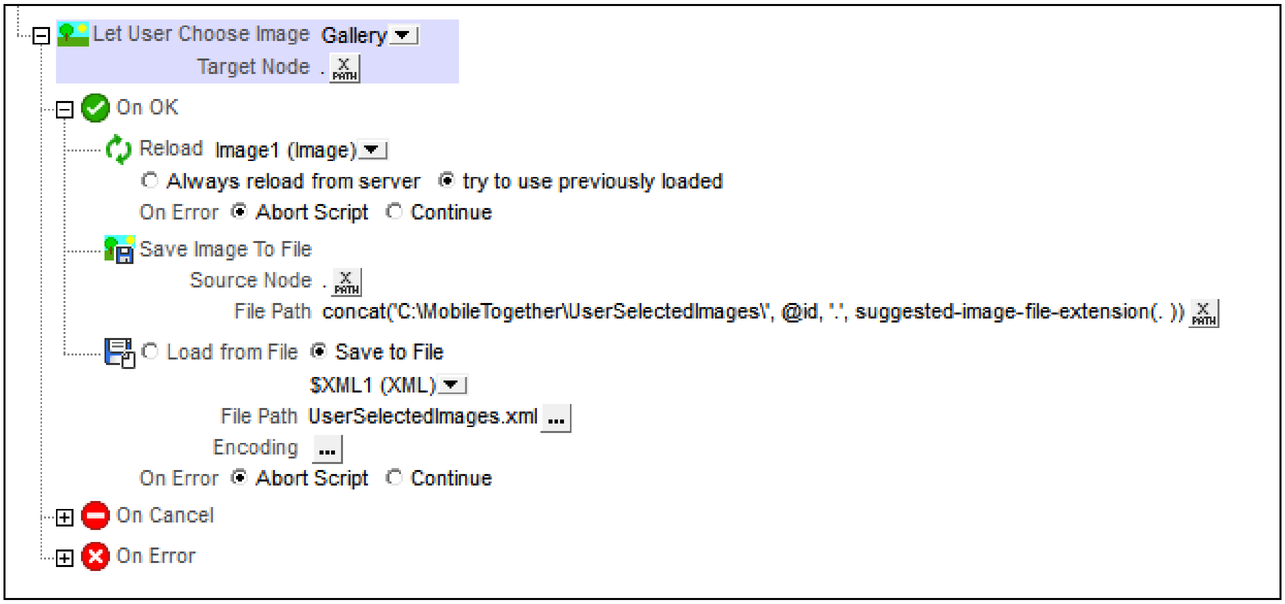
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events*[389] *and* *Control Events*[665].

## 10.2.1 Let User Choose Image

The end user can select an image that will be saved in Base64 format to a node of the page source tree (*see screenshot below*). This enables the end user to select images that are automatically saved to a database. An example of a usage scenario would be the reporting of damages for insurance purposes. The user could run the solution and take a photograph with the mobile device. The image would be directly uploaded to the appropriate database.



```
Let User Choose Image  Gallery
          Target Node  .
  On OK
    Reload  Image1 (Image)
      ○ Always reload from server  ● try to use previously loaded
      On Error  ● Abort Script  ○ Continue
    Save Image To File
          Source Node  .
            File Path  concat('C:\MobileTogether\UserSelectedImages\', @id, '.', suggested-image-file-extension(. ))
    ○ Load from File  ● Save to File
                 $XML1 (XML)
            File Path  UserSelectedImages.xml ...
             Encoding  ...
      On Error  ● Abort Script  ○ Continue
  On Cancel
  On Error
```

The action has the following properties:

- *Image source:* Select `Gallery` to let the user choose an image from the image gallery of the client device. Select `Camera` to start the camera app of the mobile device and capture the next photo taken by the camera.
- *Target Node:* A page source node in which to save the image as Base64-encoded data.

The *Let User Choose Image* action has three conditions:

- *On OK:* Define actions to perform if the image is correctly imported to target node as Base64-encoded data. Typical actions to perform would be: (i) Reload[801] the image control that displays the selected image; this updates the display with the selected image; (ii) Save Image to File[707] if the image is required to be saved as a binary image file (as opposed to being saved in an XML node as Base64-encoded text); (iii) Load/Save to File[809] saves the XML data, including the newly added Base64-encoded image data to the page source.
- *On Cancel:* If the image selection process is canceled by the user, some actions might be needed to rollback modifications made preparatory to importing the user-selected image.

- *On Error:* Define actions to take in case the image is not imported correctly into the target node. For example, the user can be informed that the selection has failed, and/or a Troubleshooting page can be opened in a web browser.

For an example of how to use this action, see the section Images Chosen by End User 1101.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

```
mt-base64-to-hexBinary()
mt-hexBinary-to-base64()
mt-hexBinary-to-string()
mt-change-image-colors()
mt-extract-file-extension()
mt-extract-file-name()
mt-image-width-and-height()
mt-string-to-hexBinary()
mt-transform-image()
```

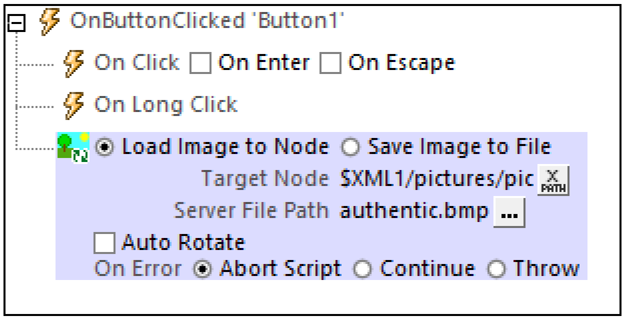
## 10.2.2 Load/Save Image

This action enables the following:

- Loading an image file into a page source node as a Base64-encoded image
- Saving a Base64-encoded image in a page source node as an image file at a server-side or other external location.

### Loading an image file into a page source node

An image file can be loaded into a page source node by using the *Load Image to Node* option of the Load/Save Image action (*see screenshot below*). Use an XPath expression to select the target node, that is, the page source node where the image data will be stored. In the *File Path* field, select the image file that is to be loaded into the target node. The image file can be any standard image format (such as BMP, EXIF, GIF, JPG, or PNG). The image file data is converted into Base64 and stored as Base64 encoded data in the target node. Note that the Base64 encoding will containing information specifying the original image format.

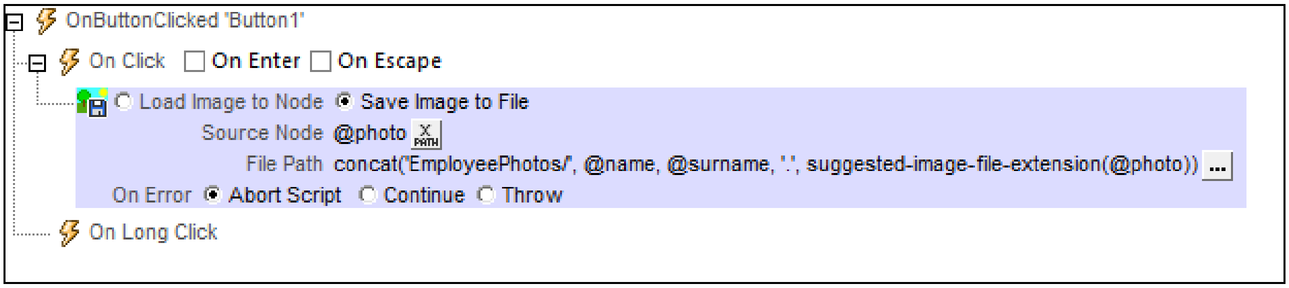


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
● Load Image to Node ○ Save Image to File
Target Node $XML1/pictures/pic
Server File Path authentic.bmp
□ Auto Rotate
On Error ● Abort Script ○ Continue ○ Throw

The *Auto Rotate* option applies to EXIF images[1092]. If it is selected, then EXIF images will be automatically rotated according to the EXIF information. The image size will be unchanged, and quality will be set at 50%. If the rotation information is not available in the EXIF file, then a rendering is nevertheless attempted and no error is reported. An error will, however, be reported if the rendering fails—for example, due to incomplete or incompatible data, or due to insufficient memory.

## Saving Base64-encoded image data as an image file

Base64 image data that is stored in a page source node can be saved as an image file by using the *Save Image to File* option of the Load/Save Image action (*see screenshot below*). Select the page source node where the Base64-encoded image is located (the *Source Node* field; *see screenshot below*). Then select the location on the server or client where the file is to be saved (the *File Path* field).



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
○ Load Image to Node ● Save Image to File
Source Node @photo
File Path concat('EmployeePhotos/', @name, @surname, '.', suggested-image-file-extension(@photo))
On Error ● Abort Script ○ Continue ○ Throw
On Long Click

When entering the path to the location where the file is to be saved, the Altova XPath extension function `suggested-image-file-extension` can be used to determine and specify the filetype of the image. Each image is of a particular image format, and this format information is stored within the Base64-encoded image data. The `suggested-image-file-extension` function returns the extension. Note that entering the wrong filetype as part of the image filename could render the image file unreadable.

The following XPath expression:

```
concat('EmployeePhotos/', @name, @surname, '.', suggested-image-file-extension(@photo))
```

would evaluate to something like this:

```
EmployeePhotos/MaxMuster.png
```

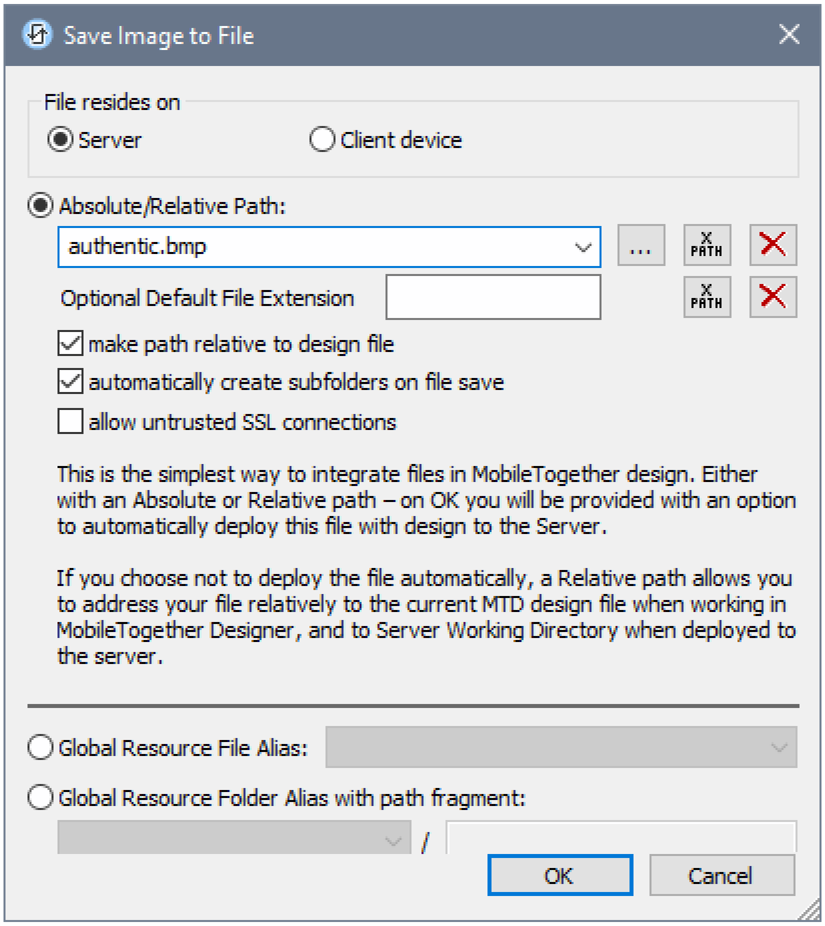
For an example of how to use this action, see the section Images Chosen by End User[1101].

## Image file locations

When you click the **Additional Dialog** button of the *File Path* field of the Load/Save Image action (*see screenshots above*), the Load Image from File dialog (for loading) or Save Image to File dialog (for saving) appears. In these dialogs, you specify whether the file is located on the server or the client by selecting the respective radio button (*see screenshots below*).

*File is located on server*

If the image file is located on the server, you can either browse for its location (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want.


Save Image to File
File resides on
● Server
○ Client device
● Absolute/Relative Path:
authentic.bmp
...
X PATH
Optional Default File Extension
☑ make path relative to design file
☑ automatically create subfolders on file save
□ allow untrusted SSL connections
○ Global Resource File Alias:
○ Global Resource Folder Alias with path fragment:
/
OK
Cancel

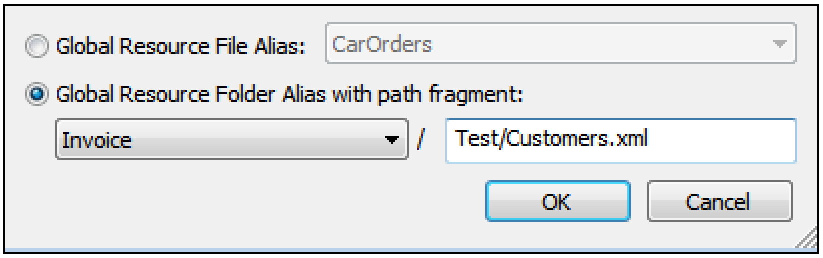
This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted

SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.
- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.
- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). See the section Altova Global Resources 1334 for details.
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

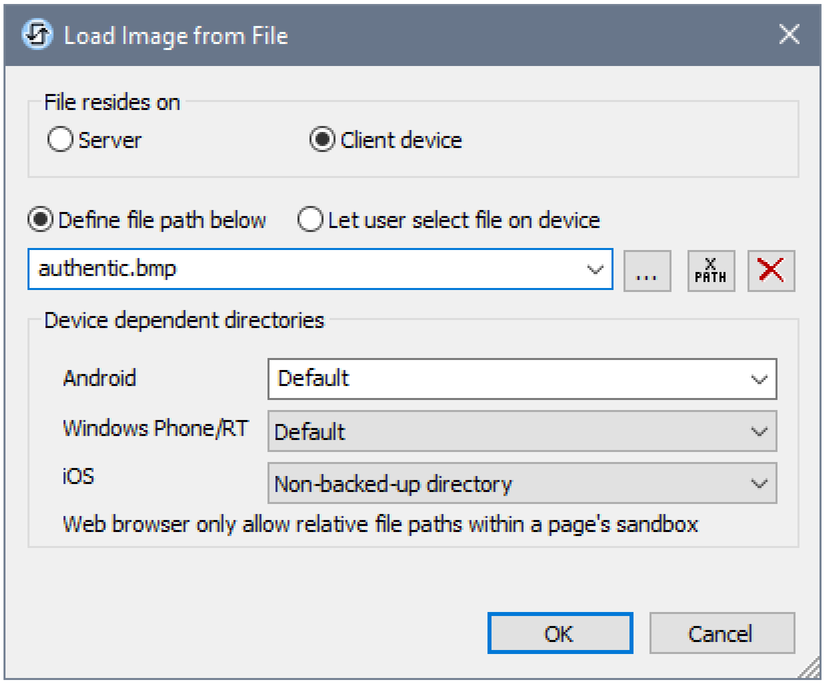
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 1334 for details.

*File is located on client*

If the image file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



Load Image from File

File resides on

○ Server ● Client device

● Define file path below ○ Let user select file on device

authentic.bmp

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To[686] (*Source File* and *Target File* options), Load/Save File[809], Load/Save Image[707], Load/Save Binary File[815], Load/Save Text File[821], Read Folder[847], and Get File Info[849].

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.
- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog[1663] (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).

- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

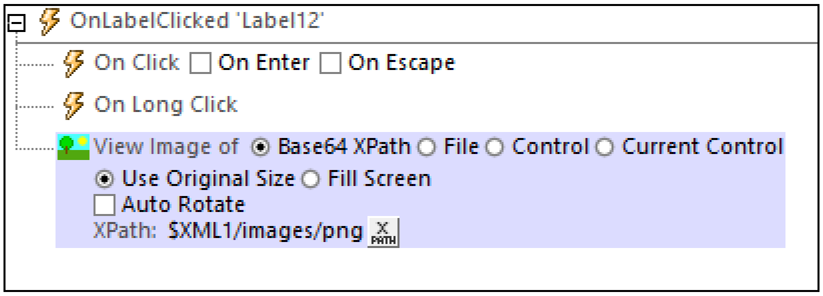
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

```
mt-last-file path()
mt-base64-to-hexBinary()
mt-hexBinary-to-base64()
mt-hexBinary-to-string()
mt-change-image-colors()
mt-extract-file-extension()
mt-extract-file-name()
mt-image-width-and-height()
mt-string-to-hexBinary()
mt-transform-image()
```

## 10.2.3 View Image

The View Image action (*screenshot below*) causes the selected image to be displayed full screen in a new view. The end user can zoom into and out of the image, and can scroll the image both horizontally and vertically.



⊟ OnLabelClicked 'Label12'

On Click □ On Enter □ On Escape

On Long Click

View Image of ● Base64 XPath ○ File ○ Control ○ Current Control

● Use Original Size ○ Fill Screen

□ Auto Rotate

XPath: $XML1/images/png

The image to display can be selected in one of the following ways:

- *Base64 XPath:* The XPath expression must either (i) contain the Base64-encoded text of the image, or (ii) select an XML page-source node that contains the Base64-encoded text.
- *File:* Enter the file path of an image file on the client machine. In the dialog box that appears, select the device directory in which the image file is located, and enter the name of the image file.
- *Control:* Select the control that holds the image. Allowed controls are Image[541], Chart[438], and Signature Field[590] controls. Instances of these controls that have been created in the design are displayed by their names in the dropdown list of a combo box. Choose the control from the dropdown list.
- *Current Control:* Shows the image of the current control. The current control should be an Image[541] or a Chart[438].

You can also select the following options:

- Whether the image should be displayed at original size or should fill the device screen. The fill-screen option enlarges or reduces the image size so that it fits the screen.
- Whether the image will be rotated according to its EXIF orientation information.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-base64-to-hexBinary()`**
**`mt-hexBinary-to-base64()`**
**`mt-hexBinary-to-string()`**
**`mt-change-image-colors()`**
**`mt-image-width-and-height()`**
**`mt-string-to-hexBinary()`**
**`mt-transform-image()`**

## 10.2.4 Scan/Generate Barcode

You can set this action to do one of the following:

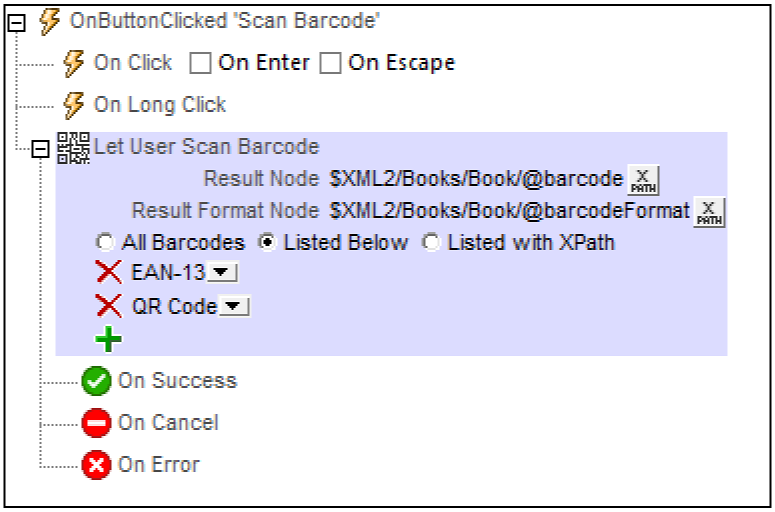
- scan a barcode with the camera of the client device and store the barcode data in a page source node, or
- generate a barcode and store the barcode image in a page source node.

Each of these actions is described separately below.

### Let User Scan Barcode

To enable the end user to scan a barcode and store the data contained in the barcode, select the option *Let User Scan Barcode (see screenshot below)*.

When the action is triggered, the camera application of the client device is launched and the end user can scan a barcode. On completing the scan, MobileTogether enters the information contained in the barcode and the corresponding barcode format into two separate page source nodes. For example, if an ISBN barcode is scanned, then the ISBN number and the ISBN barcode format (which is EAN-13) are saved, respectively, to the two nodes specified in the definition of the action (*Result Node* and *Result Format Node*, respectively). This barcode information is then available to the design as XML data.

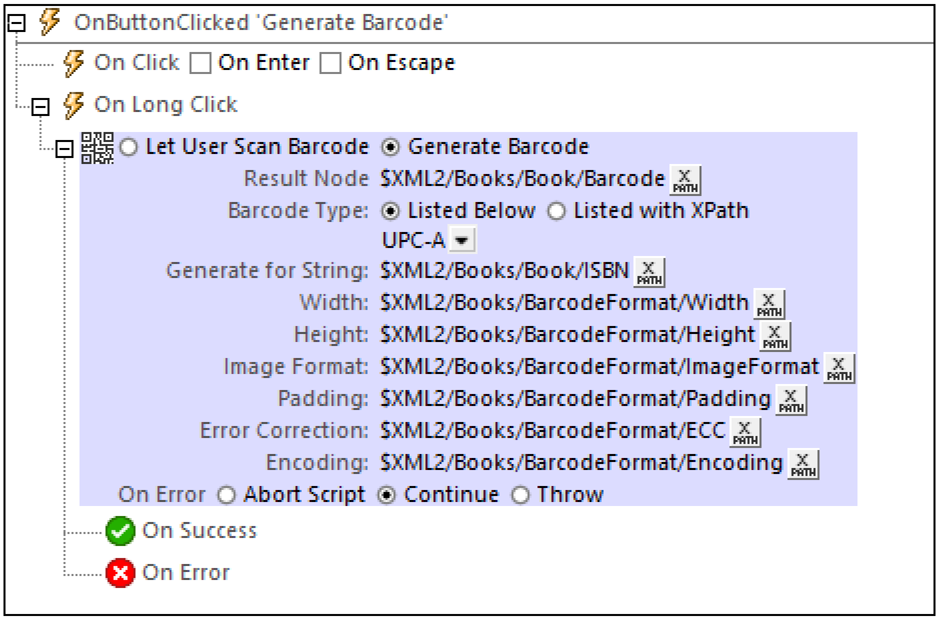

OnButtonClicked 'Scan Barcode'
On Click □ On Enter □ On Escape
On Long Click
Let User Scan Barcode
Result Node $XML2/Books/Book/@barcode
Result Format Node $XML2/Books/Book/@barcodeFormat
○ All Barcodes ● Listed Below ○ Listed with XPath
EAN-13
QR Code
On Success
On Cancel
On Error

The Let User Scan Barcode action (*see screenshot above*) has the following options:

- *Result Node:* The page source node where the barcode data obtained from the scan is saved.
- *Result Format Node:* The page source node where the format of the scanned barcode is saved. The format of the scanned barcode is detected automatically and saved to this node. The format names that are entered in this node are entered exactly as given in the list of supported formats below.
- *Barcode formats for which to scan:* Specifies what formats may be scanned. Barcodes of a format that is not in the list will not be scanned. You can select: (i) All barcodes, (ii) barcodes that you list by adding entries and then selecting a format in each entry's combo box, or (iii) a list of barcodes that is returned as an XPath sequence in which each item is a string that is a single barcode format (for example: `"Aztec", "Codabar", "Code 39"`). If you set this filter with an XPath expression, then make sure to use the name of the format exactly as it is given in the list of supported formats below.

## Generate Barcode

To generate a barcode image from barcode data and store the image in a page source node, select the option *Generate Barcode (see screenshot below)*. For example, a barcode image can be generated from an ISBN number and stored (in Base64 format) in a page source node.



OnButtonClicked 'Generate Barcode'

On Click □ On Enter □ On Escape

On Long Click

○ Let User Scan Barcode ● Generate Barcode

Result Node $XML2/Books/Book/Barcode

Barcode Type: ● Listed Below ○ Listed with XPath

UPC-A

Generate for String: $XML2/Books/Book/ISBN

Width: $XML2/Books/BarcodeFormat/Width

Height: $XML2/Books/BarcodeFormat/Height

Image Format: $XML2/Books/BarcodeFormat/ImageFormat

Padding: $XML2/Books/BarcodeFormat/Padding

Error Correction: $XML2/Books/BarcodeFormat/ECC

Encoding: $XML2/Books/BarcodeFormat/Encoding

On Error ○ Abort Script ● Continue ○ Throw

On Success

On Error

The Generate Barcode action (*see screenshot above*) has the following options:

- *Result Node:* The page source node where the generated barcode-image data is saved. The image data is saved in Base64 format.
- *Barcode Type:* The format of the barcode to generate. You can select the format from the combo box or enter it as an XPath expression. If you use an XPath expression, then make sure to specify the format exactly as given in the list of supported formats below.
- *Generate for String:* The string that is to be used as the source data for the barcode generation. For example, if you want to generate an ISBN barcode, then this string will be an ISBN number.
- *Width, Height:* The desired dimensions. Note, however, that the generated image may be larger if the dimension/s you specify are too small.
- *Image Format:* Choose either PNG or JPG as the format of the barcode image.
- *Padding:* The empty (white) space to be added on all sides of the barcode image.
- *Error Correction:* Applies to the Aztec, PDF 417, and QR Code barcode formats only. Enter a value of **`0`** to **`8`** to add additional error correction to the generated barcode. A higher value specifies greater error correction..
- *Encoding:* Applies to the Aztec, PDF 417, and QR Code barcode formats only. The encoding codepage to which the input data string is converted before barcode generation..

## Supported barcode formats

The currently supported barcode formats are listed below, together with limitations where these apply.

| Format name | Notes |
|---|---|
| `Aztec` | |
| `Codabar` | *Not on iOS* |
| `Code 39` | |
| `Code 93` | |

| | |
|---|---|
| `Code 128` | |
| `Data Matrix` | *iOS: versions >= 8* |
| `EAN-8` | |
| `EAN-13` | |
| `EAN-128` | *Detected as Code 128* |
| `ITF` | *iOS: versions >= 8* |
| `PDF 417` | |
| `QR Code` | |
| `RSS-14` | *Not on iOS. Available only for scanning, not for generation.* |
| `RSS-Expanded` | *Not on iOS. Available only for scanning, not for generation.* |
| `UPC-A` | *iOS: Detected as EAN 13* |
| `UPC-E` | |

**Note:** No barcode support is available in web clients.

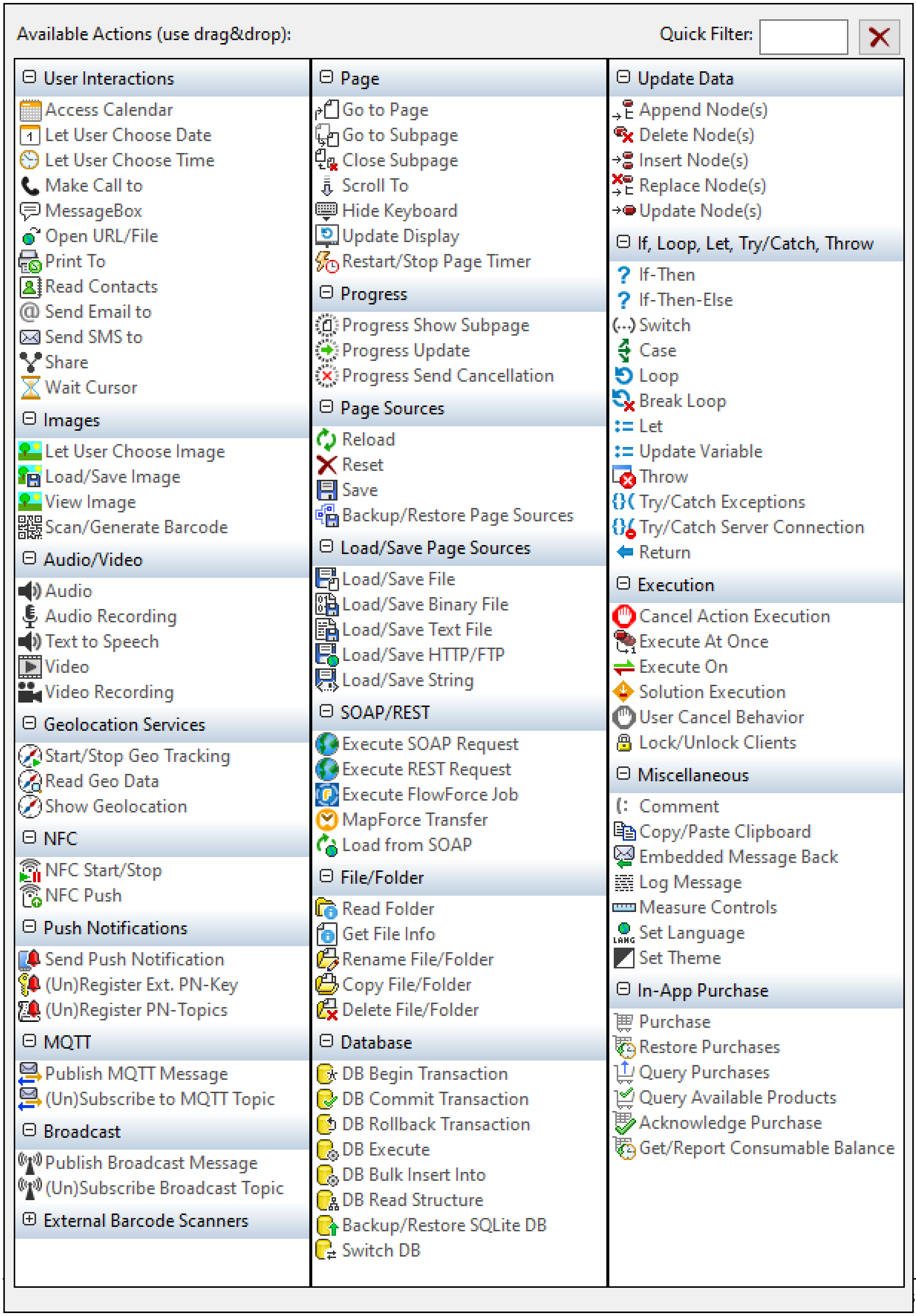
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.3 Audio/Video

The following actions are available in the Audio/Video group of the Actions dialog (*screenshot below*):

- Audio 720
- Audio Recording 1110
- Text to Speech 727
- Video 728
- Video Recording



Available Actions (use drag&drop):

Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

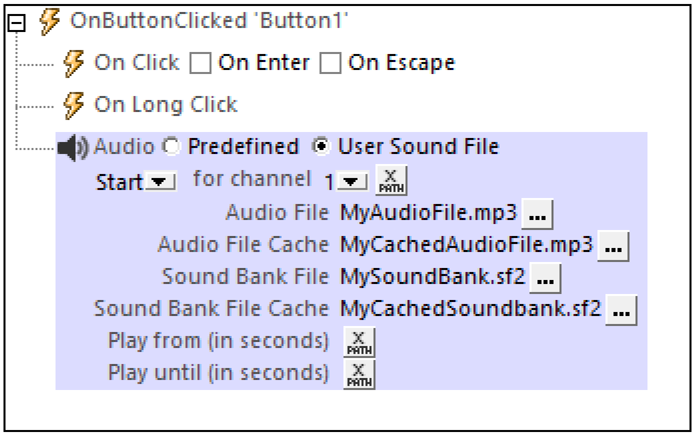
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* Page Events[389] *and* Control Events[665].

## 10.3.1 Audio

The Audio action (*screenshot below*) plays back a predefined sound or an audio file. In the action, select one of these two options, and then define the parameters of the audio playback. You can define playback for up to five different sound channels.



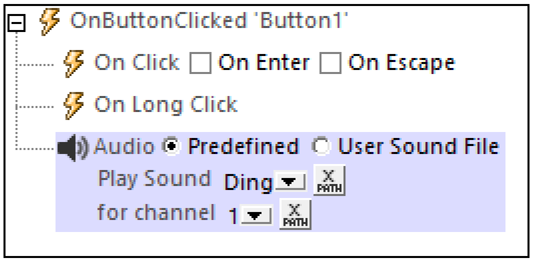
OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Audio ○ Predefined ● User Sound File
Start for channel 1
Audio File MyAudioFile.mp3
Audio File Cache MyCachedAudioFile.mp3
Sound Bank File MySoundBank.sf2
Sound Bank File Cache MyCachedSoundbank.sf2
Play from (in seconds)
Play until (in seconds)

### Predefined audio

You can choose from among the following predefined sounds, which are available on client devices:

```
ClickOffOn, ClickOnOff, Ding, DingDong, ErrorDeepBuzz, ErrorWhoops, Goodbye, KeyClickTick,
KeyClickTock, MessageBounce, MessageXylophone, WhooshDeep, WhooshExhale, WhooshLong,
WhooshQuick, WhooshQuicker
```

Either select the name of the predefined sound from the list in the combo box, or enter an XPath expression that evaluates to one of these predefined names. The names must exactly match the names in the previous paragraph.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Audio ● Predefined ○ User Sound File
Play Sound Ding
for channel 1

## Using selected sound files

The Audio action also enables the playback of a sound file that you select. In the action's dialog, you can select one of the following actions: *Start Audio, Pause, Resume, Stop, Seek (Jump) To*.

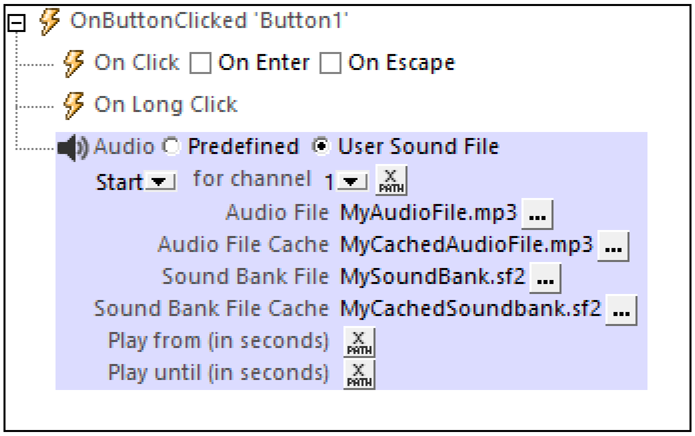
Audio can be played back on five channels (numbered **1** to **5**), and each Audio action is defined for one specific channel. So, for example, you could define the following sequence of actions:

1. Start audio playback on Channel 1
2. Pause audio playback on Channel 1
3. Start audio playback on Channel 2
4. Stop audio playback on Channel 2
5. Resume audio playback on Channel 1
6. Stop audio playback on Channel 1

Typically, the Audio action will be defined on a control event such as a button click (*see screenshot below*). The event triggers the associated Audio action. The various Audio actions are described below. For an overview of using audio in solutions, see the section Audio Playback 1108.

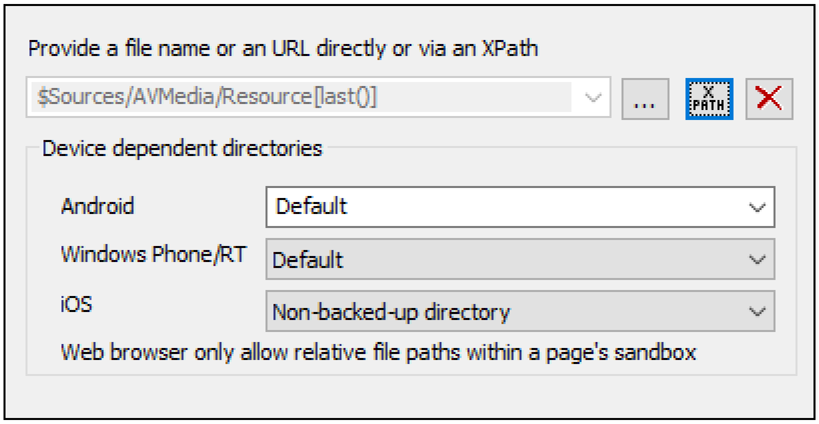
## Start audio

The Start audio action starts download and playback of the specified audio file on the specified channel. It takes the settings shown in the screenshot below.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Audio ○ Predefined ● User Sound File
Start for channel 1
Audio File MyAudioFile.mp3
Audio File Cache MyCachedAudioFile.mp3
Sound Bank File MySoundBank.sf2
Sound Bank File Cache MyCachedSoundbank.sf2
Play from (in seconds)
Play until (in seconds)

- *Audio File:* Click the **Additional Dialog** button to specify the audio file to play. The setting can be a URL or a relative file path, and can be either a static value or be located or generated by means of an XPath expression. Relative file paths are used to locate files that are stored on the mobile device. They are resolved relative to the base directory you specify for that device type (*see screenshot below*). For information on audio file formats, see Audio/Video Formats 1115.



Provide a file name or an URL directly or via an XPath

$Sources/AVMedia/Resource[last()]

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

- *Android:* Select the Android device-directory from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. Note, however, that, unless the Android device is rooted, no other app (besides MobileTogether) will be able to access the MobileTogether sandbox directory. So, trying to open a file in the MobileTogether sandbox with another app could fail.
- *Windows RT:* Select the Windows Phone or Windows RT device folder from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected.
- *iOS:* MobileTogether creates two directories on the user's device: (i) a *Backed-up directory* for files that are backed up by the operating system and will be available at later times, for example, after a restore. This directory is intended for files that are important for the user and that should not be lost; (ii) a *Non-backed-up directory* for files that do not need to be backed up, or if faster buffering is needed for playback. Select *Backed-up directory* or *Non-backed-up directory* as required.
- *Web browser:* No selection is available. Relative paths are resolved within the context of the browser's sandbox.
- If a value for the *Audio File Cache* setting is specified, then the downloaded file is cached to the designated local file. If the cache file already exists, the cache file will be played and no download takes place. Note that you must specify the full name of the cache file, including its extension, and the full file name (including the extension) must match the file name of the source file that is to be played. Note also that no advantage is gained if a cache file is specified for an audio file that is already located on the client device.
- A sound bank file is required on iOS devices to play MIDI files. The *Sound Bank File* settings specify the locations of the sound bank file and its cache. If a cached sound bank file already exists, then the cache file will be used.
- To play a segment of the audio file, enter the segment's *From* and *To* times in seconds. If you leave these fields blank, the audio file will play from start to end.

**Note:** Multi-channel audio/video playback is not supported on Windows Phone. Only one audio **or** video file can be played at a time: this is the file that was started last.

**Note:** Audio and video files **cannot** be deployed to MobileTogether Server via the MobileTogether Designer project's Deploy to Server mechanism 291. You can, however, copy audio/video files manually to the server, although you cannot stream them from there via a URL. If you wish to stream audio/video files that are located on your MobileTogether Server, then do the following: (i) use the Load Binary 815 action to load the binary audio/video data to a page source node; (ii) use the Save Binary 815 action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 1108 to play the file that is now saved on the client device. Alternatively, you can save audio/video files to a web server—instead of saving to MobileTogether Server—and use a URL to stream the audio/video file from the web server.

## Pause

This action is defined for a specific audio channel. When the action is triggered, the audio that is currently playing on the specified channel is paused. Typically, the action would be defined on a **Pause Audio** button. Note that this action applies across the project, and so would apply also to an audio file that was started on another page.

**Note:** If an audio stream is playing when a solution is suspended 915, then playback is paused. Playback continues when the solution is resumed.

## Resume

This action is defined for a specific channel, and resumes playback on that channel if it was previously paused. Typically, the action would be defined on a **Resume Audio** button. Note that this action applies to paused playback on the specified channel across the entire project, regardless of the page on which the audio file was started or paused.

**Note:** If an audio stream is playing when a solution is suspended 915, then playback is paused. Playback continues when the solution is resumed.

## Stop

When the Stop action is triggered, the audio that is currently playing on the specified channel is stopped. Typically, the action would be defined on a **Stop Audio** button, and it would apply to the audio file playing on the specified channel even if that file was started on another page.

## Seek To

Causes playback to jump to the specified position (given in seconds) of the audio file playing on the specified channel. The action applies to the audio file playing on the specified channel regardless of the page on which playback was started.

## Additional points

The section Audio (Playback) 1108 provides an overview of the audio playback feature of MobileTogether. There, you will also find information about Audio (playback) events.

## MobileTogether extension functions

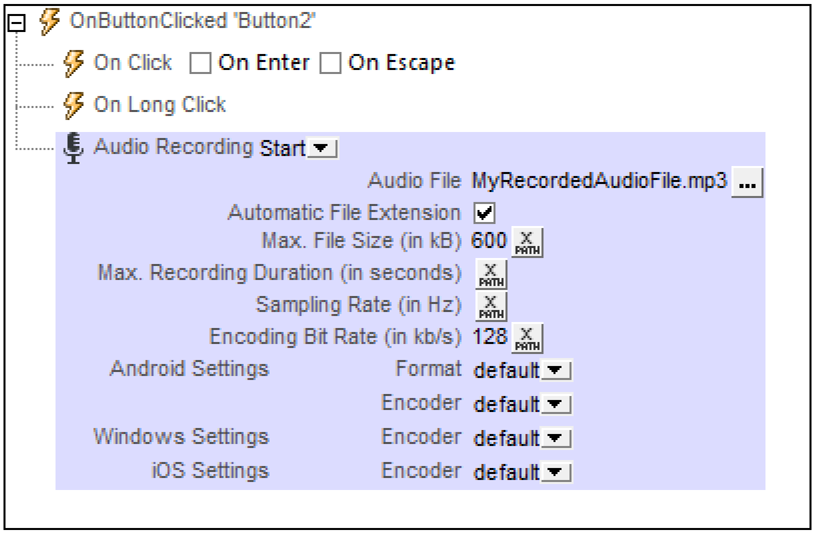
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-audio-get-current-position()`**
**`mt-audio-get-duration()`**
**`mt-audio-is-playing()`**
**`mt-audio-is-recording()`**
**`mt-extract-file-extension()`**
**`mt-extract-file-name()`**

## 10.3.2 Audio Recording

The Audio Recording action (*screenshot below*) starts or stops an audio recording. The recorded file is saved on the client device.

For example: If an *Audio Recording Start* action is associated with a button click, then the button click starts recording (via the device's mic) to the file specified in the action (*see screenshot below*); other recording parameters are also specified in the action's settings. If an *Audio Recording Stop* action is associated with a button click, then clicking this button stops any recording that was started from that page and is currently in progress.



OnButtonClicked "Button2"
On Click □ On Enter □ On Escape
On Long Click
Audio Recording Start
Audio File MyRecordedAudioFile.mp3
Automatic File Extension ☑
Max. File Size (in kB) 600
Max. Recording Duration (in seconds)
Sampling Rate (in Hz)
Encoding Bit Rate (in kb/s) 128
Android Settings Format default
Encoder default
Windows Settings Encoder default
iOS Settings Encoder default

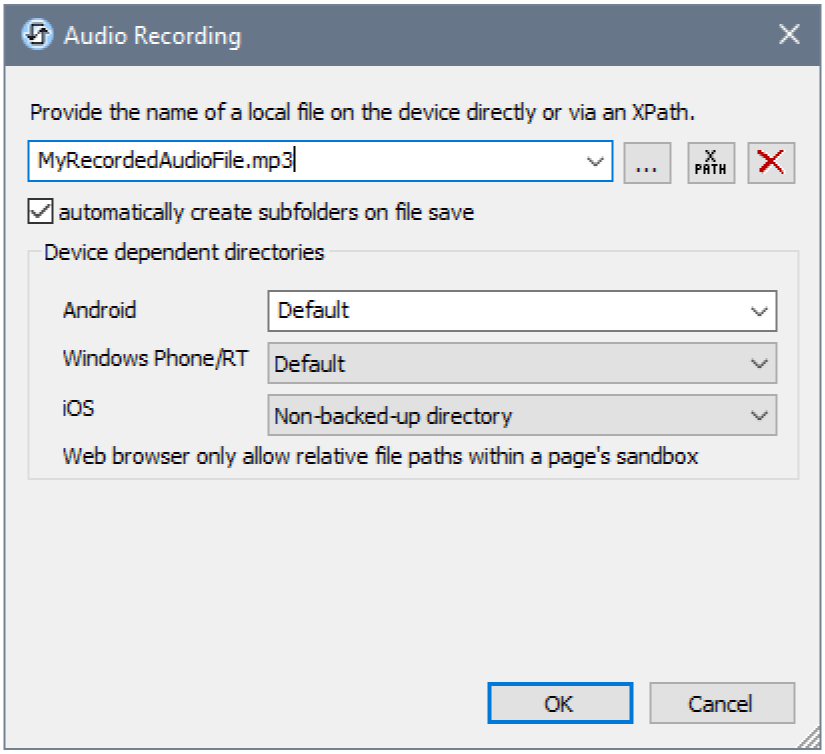
**Note:** If an audio recording is in progress when a solution is suspended[915], then the recording is stopped.

**Note:** Audio should not be recorded at the same time as audio/video is being played back as this could result in problems with the playback state, particularly on iOS devices.

### Start audio recording

In the Audio Recording combo box (*see screenshot*) select *Start*.

For the *Audio File* setting, provide the relative path and name of the client-device file in which you want to save the recording. Do this by either entering the file path directly or selecting the file path via an XPath expression. The file path you provide will be resolved relative to the base directory you specify for that device type (*see screenshot below*). If the path contains folders that do not exist, you can check the option to automatically create missing sub-folders (*see screenshot below*). For information about audio file formats, see the section Audio/Video Formats[1115].



Audio Recording

Provide the name of a local file on the device directly or via an XPath.

MyRecordedAudioFile.mp3

☑ automatically create subfolders on file save

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web browser only allow relative file paths within a page's sandbox

- *Android:* Select the Android device-directory from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. Note, however, that, unless the Android device is rooted, no other app (besides MobileTogether) will be able to access the MobileTogether sandbox directory. So, trying to open a file in the MobileTogether sandbox with another app could fail.
- *Windows RT:* Select the Windows Phone or Windows RT device folder from the dropdown list. If you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected.
- *iOS:* MobileTogether creates two directories on the user's device: (i) a *Backed-up directory* for files that are backed up by the operating system and will be available at later times, for example, after a restore. This directory is intended for files that are important for the user and that should not be lost; (ii) a *Non-backed-up directory* for files that do not need to be backed up, or if faster buffering is needed for playback. Select *Backed-up directory* or *Non-backed-up directory* as required.
- *Web browser:* No selection is available. Relative paths are resolved within the context of the browser's sandbox.

Next, set up the parameters of the recording (*see screenshot at the top of the page*):

- *Automatic File Extension:* If selected, the file extension will be automatically added depending on the selected format and/or codec. You can use the XPath extension function `mt-extract-file-extension` 1262 to retrieve the extension of the last audio file that was recorded.
- *Max File Size (in kB):* The default value is unlimited. Note that the file size depends on the sampling rate and encoding bitrate.
- *Max Recording Duration (in seconds):* The default value is unlimited.
- *Sampling Rate (in Hz):* A higher sampling rate produces a more accurate digital representation of the audio signal. The downside of a higher sampling rate, however, is a larger file size. For your reference when selecting a sampling rate, CD-quality sampling is 44.1 kHz (44100 Hz). Note, however, that the sampling rate depends to a large extent on the encoder and audio coding standard being used. We

recommend that you leave this setting blank so that the encoder will automatically use its own default setting. If you wish to specify a sampling rate, be sure to consult the related audio encoding standard or the encoder specifications.

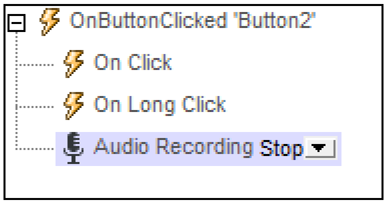
- *Encoding Bitrate (in kb/s):* Essentially the level of compression. When an audio file is compressed, details are removed. Usually these details are frequencies outside the human frequency range (20 Hz to 20 kHz) or frequencies that are reproducible only by high-end speakers. The tradeoff is between file size and audio quality. A bitrate of 128 kbps is reasonable quality and typical for MP3s. Streaming audio is usually 256 kbps. CD-quality is 320 kbps. Typical bitrate values are: 96 kbps (standard quality on mobiles); 160 kbps (high quality on mobiles; standard quality on desktops); 320 kbps (extremely high quality on mobiles; high quality on desktops). We recommend that you leave this setting blank so that the encoder will automatically use its own default setting. If you wish to specify a bitrate, be sure to consult the related audio encoding standard or the encoder specifications.
- *Android Settings:* Select from the formats and audio recording encoders (codecs) available on Android devices. *Default* selects the default format/codec on the client device. For the `AMR-NB` and `AMR-WB` formats, use only the AMR-NB and AMR-WB codecs, respectively. The `AAC` format can be generated with the AAC-Low Complexity, High Efficiency-AAC, and Enhanced Low Delay-AAC codecs, but it is possible that not all of these codecs will be available or work on a given device. Unless you are aware of the effect of your choices for solution users, we recommend that you leave both settings at `Default`. See Audio/Video Formats 1115 for more information.
- *Windows Settings:* Select from the audio recording encoders (codecs) available on Windows devices. *Default* selects the default codec on the client device.
- *iOS Settings:*Select from the audio recording encoders (codecs) available on iOS devices. *Default* selects the default codec on the client device.

An audio recording will be stopped in the following situations:

- A Stop Audio Recording action is executed (*see below*). Such an action is executed when triggered by a client-side event, such as a button-click.
- The end user leaves the page or a page leave is executed.
- The solution is suspended 915.

## Stop audio recording

Set the Audio Recording action to Stop (*see screenshot below*) to stop any audio recording that was started by an Audio Recording action on the same page and that is currently in progress.



**Note:** A recording is also stopped when a solution is suspended 915 or when the user leaves the page on which the recording was started.

## Additional points

The section Audio Recording 1110 provides an overview of the audio recording feature of MobileTogether. There, you will also find information about Audio Recording events 1110.

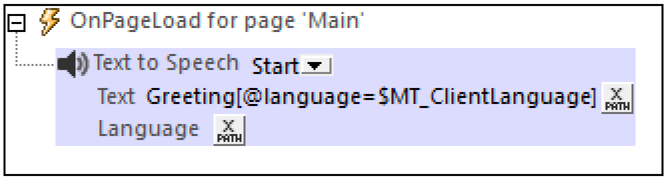
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

```
mt-audio-get-current-position()
mt-audio-get-duration()
mt-audio-is-playing()
mt-audio-is-recording()
mt-last-file path()
mt-extract-file-extension()
mt-extract-file-name()
```

## 10.3.3 Text to Speech

The Text to Speech action, when set to *Start* (*see screenshot below*), converts a text string to speech and plays it back. The text string is either entered directly in an XPath expression or is taken from a node that is selected via an XPath expression. The screenshot below envisions a situation in which, when the main page is loaded, a greeting is played back in the language of the mobile device. The text of the greeting is taken from the `Greeting` element that has a `language` attribute with a value that matches the language setting of the mobile device.



OnPageLoad for page 'Main'

Text to Speech Start

Text Greeting[@language=$MT_ClientLanguage]

Language

The action's *Language* setting is set by default to the language setting of the mobile device. It can be used to override the device's language setting. This can be required, for example, if the text string is in a specific language and the conversion has to be attempted in that specific language rather than the language of the mobile device. The value of the *Language* setting is obtained via an XPath expression. Values must be in the Language-Country format, for example: `en-US`, `en-UK`, `de-DE`, `es-US`, `fr-CH`.

**Note:** If you enter more than one Text to Speech action, then the last one in sequential order is used.

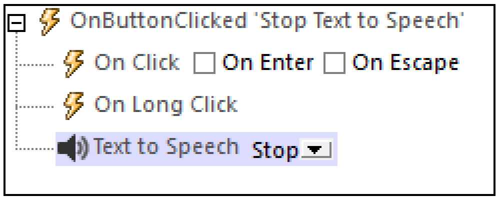
**Note:** Text to Speech playback is available on mobile devices only and cannot be simulated on MobileTogether Designer.

### Additional Text to Speech actions

Text to Speech functionality can be extended by using the additional Text to Speech actions of the project properties[296] of the design. For an overview of Text to Speech functionality, see the section Audio, Video | Text to Speech[1111].

### Stopping a Text to Speech action

While a Text to Speech playback is running, it can be stopped by using the *Stop* function of the Text to Speech action (*see screenshot below*). When the action is triggered, any Text to Speech playback that is running at that time will be stopped. No specific text needs to be specified.


⊟ OnButtonClicked 'Stop Text to Speech'
On Click □ On Enter □ On Escape
On Long Click
Text to Speech Stop

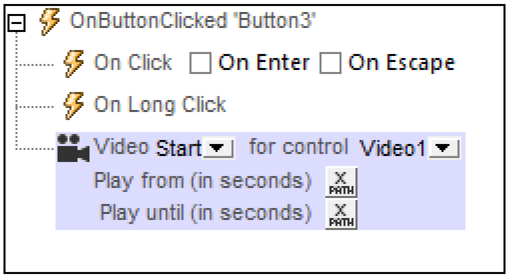
**Note:** Only one Text to Speech playback can be running at a time.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.3.4 Video

The Video action (*screenshot below*) provides one of the following actions for a Video control[655] on the current page: *Start, Pause, Resume, Stop, Seek (Jump) To*.


⊟ OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
Video Start for control Video1
Play from (in seconds)
Play until (in seconds)

All the video controls[655] on the current design page are listed by their names in the dropdown list of the *Control* combo box (*see screenshot above*). Select the video control to which you want the action to apply; in the screenshot above *Video1* has been selected.

Next, select the action you want to execute. The following actions are available:

- *Start:* To play a segment of the video, enter the segment's *From* and *To* times in seconds. If you leave these fields blank, the video will play from start to end.
- *Pause:* Pauses the video when this action is triggered. For example, the *Pause* action can be defined on a **Pause Video** button.

- *Resume:* Resumes the video when this action is triggered. It can be triggered, for example, on a button-click event.
- *Stop:* Stops the video when this action is triggered. It can be set, for example, as the action of a **Stop Video** button.
- *Seek To:* Jumps to the specified position.

**Note:** If a Video control's 655 **`Play on Load`** or **`Show Controls`** property is set to `false`, video playback can only be controlled via the actions of the Video action that have been listed above.

**Note:** Each Video control 655 also provides three events for which you can specify actions to execute: **`OnVideoStarted`**, **`OnVideoError`**, **`OnVideoCompleted`**. These event definitions can be accessed via the **Video Control Actions** command of the Video control's 655 context menu or by clicking the **Additional Dialog** button of the Video control's 655 `Control Action` property.

### MobileTogether extension functions

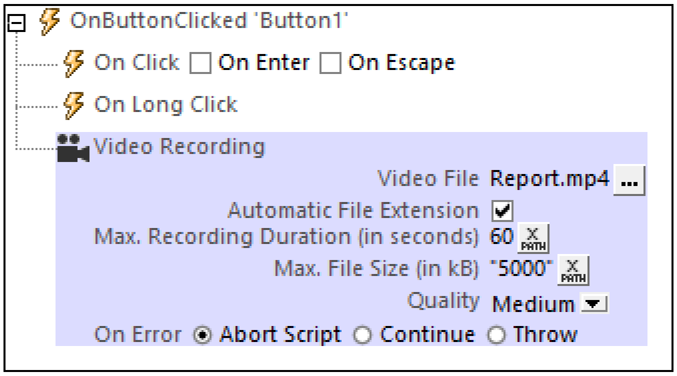
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-video-get-current-position()`**
**`mt-video-get-current-duration()`**
**`mt-video-height()`**
**`mt-video-is-playing()`**
**`mt-video-width()`**

## 10.3.5 Video Recording

The Video Recording action (*screenshot below*) opens the camera on the mobile device and enables the user to start (and then stop) recording. For example, say the action has been created for an `OnButtonClicked` event. When the relevant button is clicked in the MobileTogether solution, the client device's video recording app is opened, and the user can start/stop recording. You can specify key properties of the recording, such as: the file name and location, the maximum recording duration, the maximum file size of the recording, and the recording quality. If the video recording action is re-run on the client device, then: (i) if a file name has been specified, the new recording overwrites the previous recording; (ii) if no file name has been specified, the recording is saved as a new file.

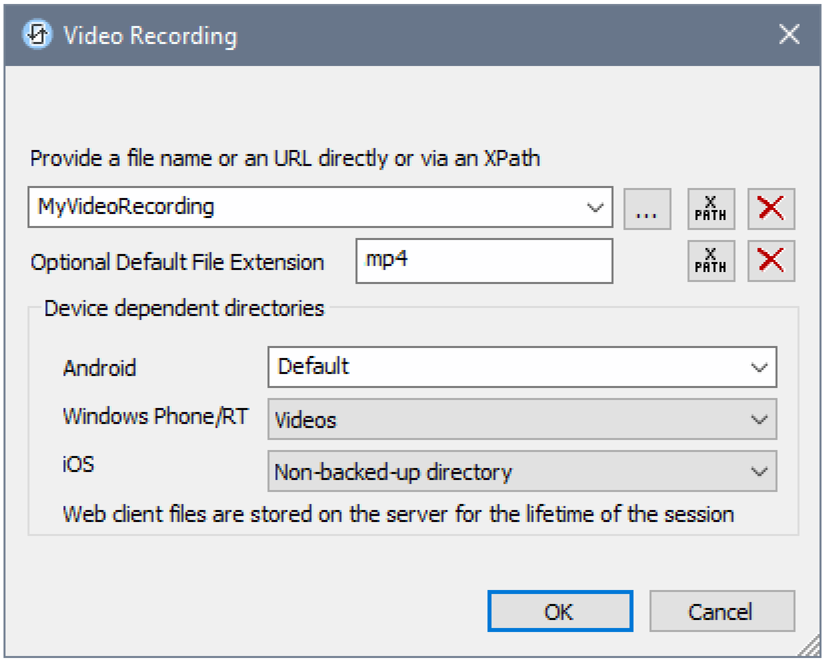
**Note:** Web clients do not support video recording.



OnButtonClicked 'Button1'
- On Click □ On Enter □ On Escape
- On Long Click
- Video Recording
  - Video File: Report.mp4
  - Automatic File Extension: ☑
  - Max. Recording Duration (in seconds): 60
  - Max. File Size (in kB): "5000"
  - Quality: Medium
  - On Error: ● Abort Script ○ Continue ○ Throw

You can define the following properties for video recordings:

- *Video File:* On clicking the property's **Additional Dialog** button, the Video Recording dialog *(screenshot below)* appears.



Video Recording

Provide a file name or an URL directly or via an XPath: MyVideoRecording

Optional Default File Extension: mp4

Device dependent directories

| | |
|---|---|
| Android | Default |
| Windows Phone/RT | Videos |
| iOS | Non-backed-up directory |

Web client files are stored on the server for the lifetime of the session

Here you can specify, either directly or via an XPath expression, the file name or URL of the recorded video file, or skip setting the file name. Note the following points:

- If the file name does not contain a file extension, then the value of the *Optional Default File Extension* setting (in the Video Recording dialog) is appended to the file name. Specify the file extension without a dot *(see screenshot above)*.
- If the file name does not contain a file extension and if no default file extension has been set (previous point), then the recording app's default file extension is appended to the file name.
- If the file name is specified without a file path, then the recording is stored relative to the directory specified in the *Device dependent directories* setting *(see screenshot above)*.
- If no file name is specified, then the recording app saves each recording with a new file name to the device's default video-recording location.

- o You can use the MobileTogether XPath extension function mt-last-file-path 1262 to get the file path of the recording, and the mt-extract-file-name 1262 function to get the file name from the submitted file path. Note that the **mt-last-file-path** function might not be supported on non-Android devices. Where supported on iOS and Windows devices, it will return the relative path from the base directory that is selected in the *Device dependent directories* setting; on Windows devices this is the `Videos` directory.

- *Automatic File Extension:* If selected, then the client device selects the file extension on the basis of the recording format or codec. If the *Optional Default File Extension* setting of the previous property has also been set, then this default file extension is used. On Windows devices, however, recordings are always saved in **.mp4** format. (Note that Windows devices will play back non-MP4 formats if these formats are associated with the media player.) The MobileTogether XPath extension function mt-extract-file-extension 1262 can be used to retrieve the file extension of a file from the the submitted file path. (A file path can be submitted with the mt-last-file-path 1262 function.)

- *Max. Recording Duration:* The recording will stop once the number of seconds specified in this setting is reached. The number that is entered can be an integer or decimal, and it can be entered as a number (without quotes) or as a string (with quotes). Valid example values are: **10.5, "10"**.
  _Note_: This setting does not work on some Android devices, and was ignored on the Windows devices that were used in Altova tests.

- *Max. File Size:* The recording will stop once the file size in kB that is specified in this setting is reached. The number that is entered can be an integer or decimal, and it can be entered as a number (without quotes) or as a string (with quotes). Valid example values are: **1000, "5000.5"**.
  _Note_: This setting does not work on Windows and iOS devices.

- *Quality:* Select a value from the combo box. Note that the higher the quality, the greater the file size will be.

_Error processing_

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

## MobileTogether extension functions

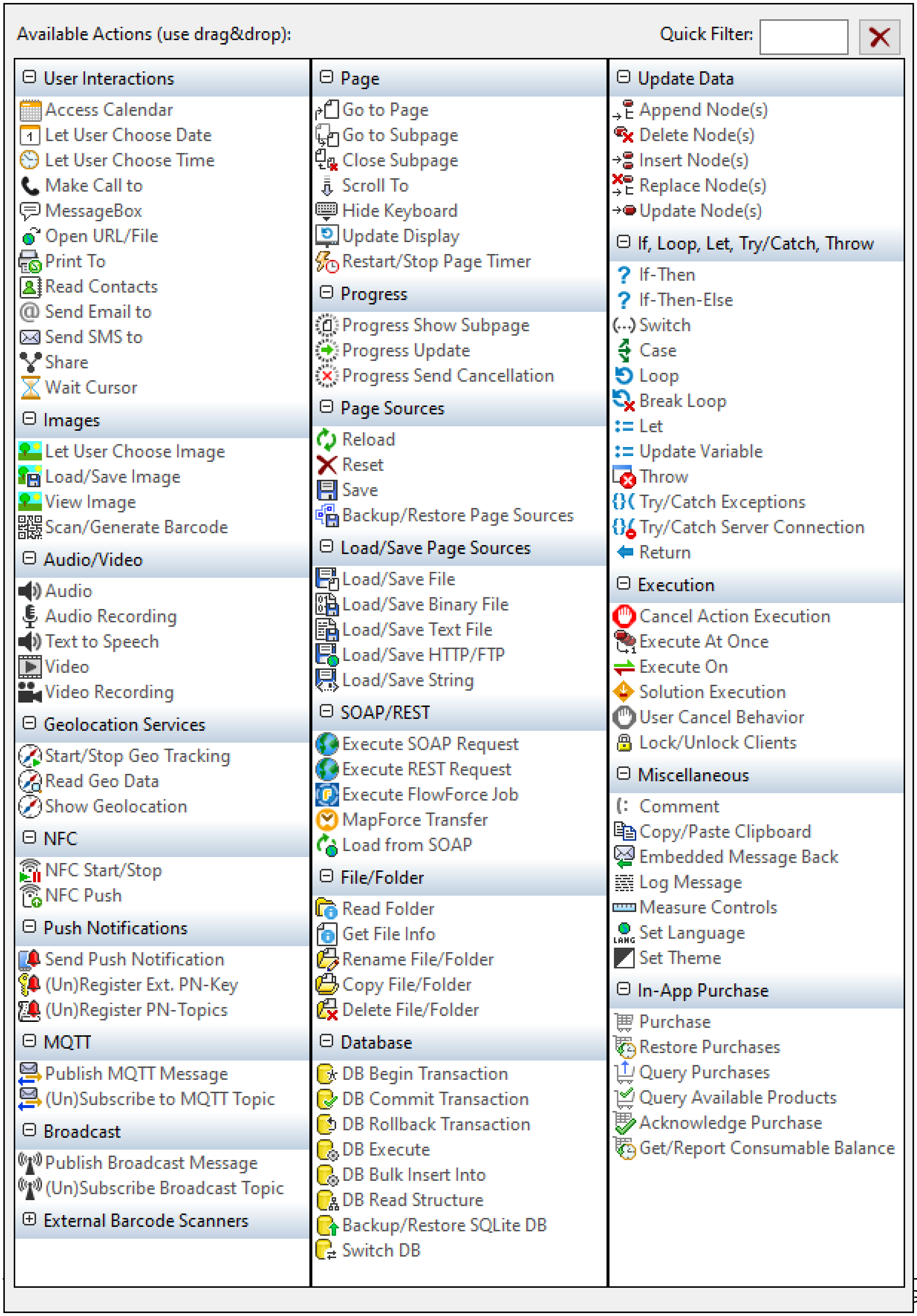
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-last-file path()`**
**`mt-extract-file-extension()`**
**`mt-extract-file-name()`**
**`mt-video-get-current-position()`**
**`mt-video-get-current-duration()`**
**`mt-video-height()`**
**`mt-video-is-playing()`**
**`mt-video-width()`**

# 10.4 Geolocation Services

The following actions are available in the Geolocation group of the Actions dialog (*screenshot below*):

- Start/Stop Geo Tracking 735
- Read Geo Data 736
- Show Geolocation



Available Actions (use drag&drop):

Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

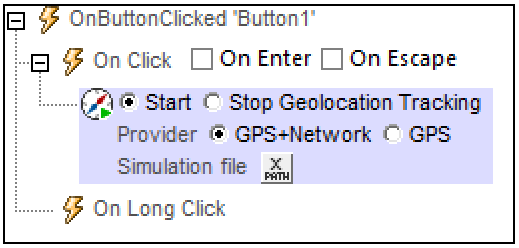
⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events*[389] *and Control Events*[665].

The tutorial Sharing Geolocations[230] shows how the Start/Stop Geo Tracking[735] and the Read Geo Data[736] actions can be used.

## 10.4.1 Start/Stop Geo Tracking

The Start/Stop Geo Tracking action (*screenshot below*) starts and stops tracking the geolocation of the mobile device. After having been started, tracking continues till stopped. Every time when the Read Geo Data[736] action is executed, the geolocation data at that moment in time is read from the mobile device, and is entered into the `$MT_GEOLOCATION` tree. The data in the `$MT_GEOLOCATION` tree can then be accessed with an XPath expression.



⊟ OnButtonClicked 'Button1'

⊟ On Click □ On Enter □ On Escape

● Start ○ Stop Geolocation Tracking

Provider ● GPS+Network ○ GPS

Simulation file

On Long Click

The tutorial Sharing Geolocations[230] shows how the Start/Stop Geo Tracking[735] action can be used.

### Start geo tracking

To start geolocation tracking on the mobile device, select the action's *Start* radio button (*see screenshot above*).

Select the geolocation information source that is best suited to the solution:

- *GPS+Network:* If the mobile device can be located with GPS, then GPS is used to provide geolocation data. Otherwise, the network provider's geolocator mechanism is used. Geolocation from the network is often less accurate than GPS, but this option has the advantage that geolocation information is always provided. The network thus acts as a backup information source if GPS is not available (for example, inside buildings).
- *GPS:* GPS is used to provide geolocation data. The advantage of using this option is that geolocation data will be accurate. The disadvantage is that, if GPS is not available at a particular location (for example, inside buildings), then no geolocation data will be available.

**Note:** The `$MT_GEOLOCATION` tree is automatically added to the page sources of the page when the Start/Stop Geo Tracking[735] action or the Read Geo Data[736] action is added to design.

*Geolocations for simulations*

For designer[1356] and server[1362] simulations, you can simulate a geolocation by specifying a geolocations XML file[1372] to use. The XPath expression that selects this file the must resolve to a URL that locates the file. The

URL can be absolute, or one that is relative to the design file. If no geolocations XML file[1372] is specified with this action, then the default geolocations file[1372] defined in the Geolocation Settings dialog[1372] is used.

### Stop geo tracking

To stop geolocation tracking on the mobile device, select the action's *Stop* radio button (*see screenshot above*).

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-change-image-colors()`**
**`mt-geolocation-started()`**

## 10.4.2 Read Geo Data

The Read Geo Data action enters the current geolocation data into the `$MT_GEOLOCATION` tree. In order for the action to be able to read the current geolocation, the mobile device's geolocation tracking must have been started[735] before this action is executed.

The `$MT_GEOLOCATION` tree is automatically added to the page sources of the page when the Start/Stop Geo Tracking[735] action or the Read Geo Data[736] action is added to design. The `$MT_GEOLOCATION` tree has two parts: `Location` and `Address` (*see listings below*). The **`Location`** element contains the geolocation coordinates. The **`Address`** element contains the address equivalent plus other details of the geolocation coordinates as determined by a directory look-up. If no postal address equivalent is available, then this part of the tree will not be filled; other child elements of `Address` (such as `URL`) might also not be filled if the relevant data is not available.

```
$MT_GEOLOCATION
<Root>
  <Location/>
  <Address/>
</Root>
```

⊟ *Detailed structure of the $MT_GEOLOCATION tree*

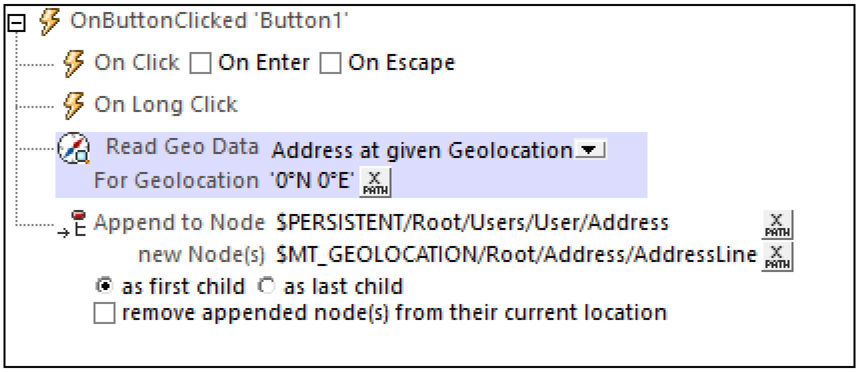
```
$MT_GEOLOCATION
<Root>
  <Location
    Provider=""
    Latitude=""
    Longitude=""
    Geolocation=""
    Altitude=""
    AccuracyVertical=""
```

```
        AccuracyHorizontal=""
        Speed=""
        Time=""
        MagneticHeading=""
      />
      <Address
        Locality=""
        SubLocality=""
        CountryName=""
        CountryCode=""
        PostalCode=""
        AdminArea=""
        SubAdminArea=""
        FeatureName=""
        Thoroughfare=""
        SubThoroughfare=""
        Phone=""
        Url=""
        Premises="">
        <AddressLine></AddressLine>
         ...
        <AddressLine></AddressLine>
      </Address>
    </Root>
```

## Geolocation retrieval options

In the dropdown box of the action's setting, you can select one of the following:

- *Current Geolocation:* Enters the mobile device's geolocation data in the `Location` element of the `$MT_GEOLOCATION` tree. Only the tree's `Location` element attributes will therefore contain data. The tree will not have an `Address` element.
- *Current Geolocation + Address:* Enters data into both the `Location` and `Address` element nodes.
- *Address at Given Geolocation:* Enters `Address` element data in the `$MT_GEOLOCATION` tree. This data corresponds to the *For Geolocation* coordinates you enter. The *For Geolocation* coordinates must be entered as a string having one of the lexical formats described in the *Geolocation input string formats* section below.The address data is obtained by looking up a geolocation directory.
- *Geolocation at Given Address:* Geolocation coordinates are fetched for the string that you enter as the value of the *For Address* field. This string is looked up in a geolocation directory, and if coordinates for this address are available, then the `Location` element of the `$MT_GEOLOCATION` tree is updated with these coordinates. You can enter any sub-part of the address for the directory lookup.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Read Geo Data Address at given Geolocation
For Geolocation '0°N 0°E'
Append to Node $PERSISTENT/Root/Users/User/Address
new Node(s) $MT_GEOLOCATION/Root/Address/AddressLine
● as first child ○ as last child
□ remove appended node(s) from their current location

*Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (") while unit indicators that are escaped are highlighted in blue ("").

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example*: **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example*: **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example*: **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example*: **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example*: **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example*: **`33.33  -22.22`**

*Examples of format-combinations:*

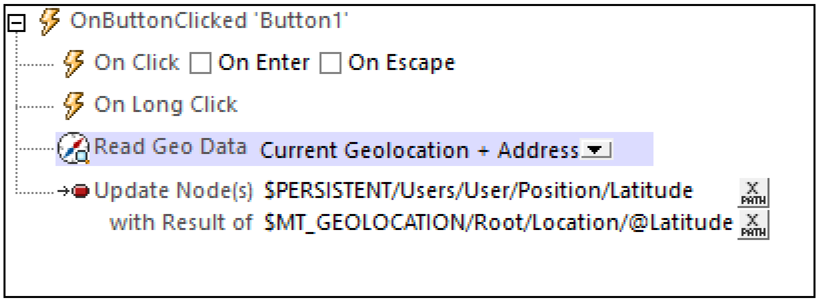
**`33.33N  -22°44'55.25"`**
**`33.33  22°44'55.25"W`**

```
33.33  22.45
```

- *Geolocation at Given Address:* Returns the geolocation of the address submitted in the *For Address* option. The address is entered as a string, for example: `"Address Line 1, Address Line 2"`. This string is submitted for a geolocation lookup, and the returned geolocation data components are stored in the `$MT_GEOLOCATION` tree (*see the tree structure listing at the beginning of the section*[736]).

## Usage

In order to use geolocation data, it must first be entered in the `$MT_GEOLOCATION` tree with the Read Geo Data action. The screenshot below, for example, shows a Read Geo Data action that enters data for both the `Location` and `Address` elements. It then accesses the `Location/@Latitude` data in the `$MT_GEOLOCATION` tree to update a node in another tree.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Read Geo Data Current Geolocation + Address
Update Node(s) $PERSISTENT/Users/User/Position/Latitude
with Result of $MT_GEOLOCATION/Root/Location/@Latitude

### *Units and datatypes of retrieved geolocation data*

The geolocation data that is retrieved from the different mobile devices is placed in the `$MT_GEOLOCATION` tree as numbers. The units and dataypes of these numbers are given in the table below.

| | Android | Web | iOS | Windows Phone | WindowsRT |
|---|---|---|---|---|---|
| **`Latitude`** | Degrees (*as double*) | Decimal degrees (*as double*) | Degrees (*as double*) | Degrees (*as double*) | Degrees (*as double*) |
| **`Longitude`** | Degrees (*as double*) | Decimal degrees (*as double*) | Degrees (*as double*) | Degrees (*as double*) | Degrees (*as double*) |
| **`Accuracy`** | Meters (*as double*) | Meters (*as double*) | Meters (*as double*) | Meters (*as double*) | Meters (*as double*) |
| **`Altitude`** | Meters above WGS 84 ref ellipsoid | Meters (*as double*) | Meters (*as double*) | Meters (*as double*) | Meters (*as double*) |
| **`Speed`** | Meters/second (*as double*) | Meters/second (*as double*) | Meters/second (*as double*) | Meters/second (*as double*) | Meters/second (*as double*) |
| **`Time`** | UTC time | DOM TimeStamp (*unsigned long long*) | NSDate (*can be converted to TZ*) | Int64/`System.Date timeOffset` (UTC) | Long long (UTC) |

For information about specifying geolocation data for designer and server simulations, see the section Geolocation Settings[1372].

The tutorial Sharing Geolocations[230] shows how the Read Geo Data[736] action can be used.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

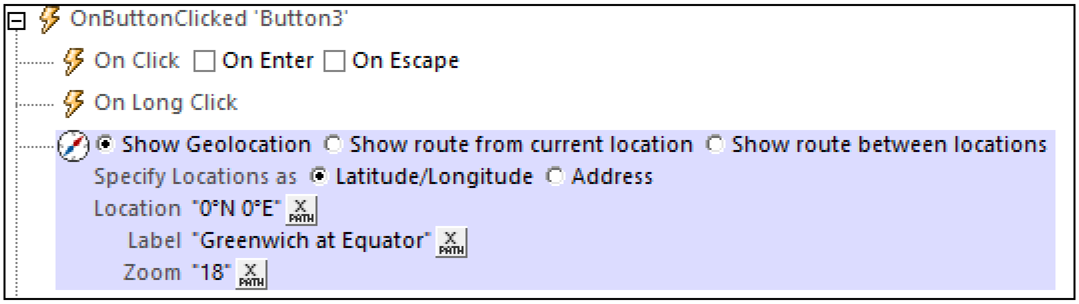
**`mt-geo-map-marker()`**
**`mt-geolocation-started()`**

## 10.4.3 Show Geolocation

The Show Geolocation action opens the map application of the mobile device and displays the specified information. You can specify one of the following sub actions (*see screenshot below*):

- *Show geolocation:* The location on the map of the coordinates that you specify
- *Show route from current location:* The route on the map from the current location to a specified location
- *Show route between locations:* The map route on the map between two specified locations

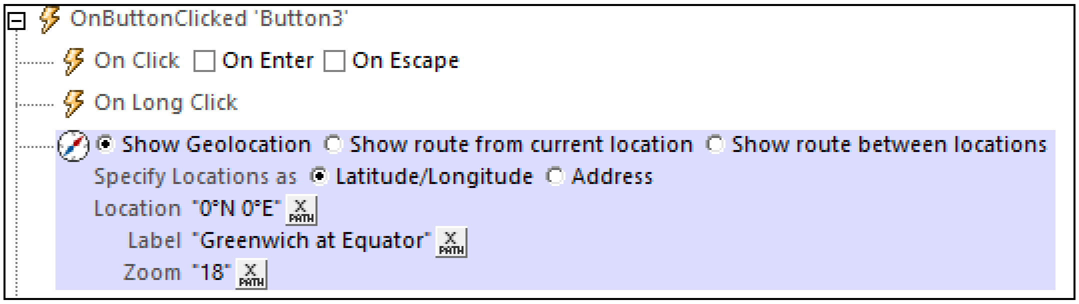
All locations in these options are given by their geolocation coordinates[742].



```
⊟ OnButtonClicked 'Button3'
    On Click □ On Enter □ On Escape
    On Long Click
    ● Show Geolocation ○ Show route from current location ○ Show route between locations
      Specify Locations as ● Latitude/Longitude ○ Address
      Location "0°N 0°E"
         Label "Greenwich at Equator"
         Zoom "18"
```

### Show geolocation

The *Show Geolocation* action opens the map application of the mobile device, and places a pin and label at the specified geolocation.



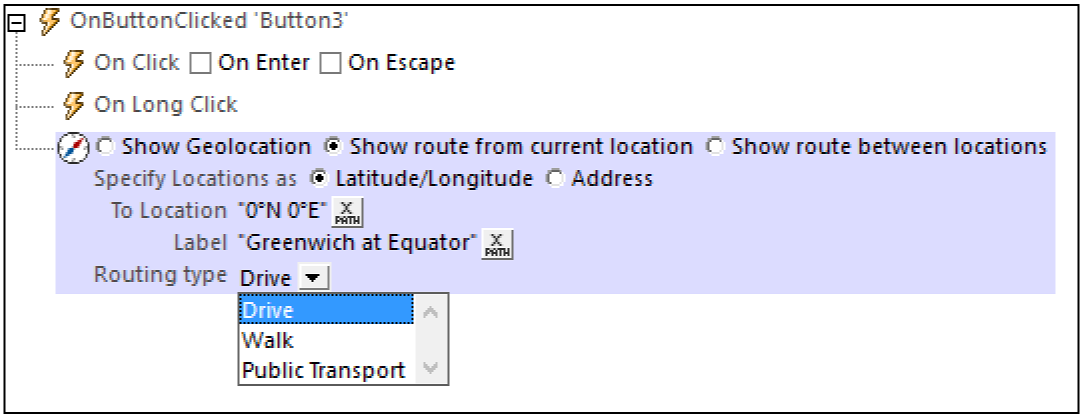
OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
● Show Geolocation ○ Show route from current location ○ Show route between locations
Specify Locations as ● Latitude/Longitude ○ Address
Location "0°N 0°E"
Label "Greenwich at Equator"
Zoom "18"

The *Show Geolocation* action has the following settings:

- *Specify Location as:* Whether the entry of the *Location* setting (next setting) is a latitude/longitude coordinate or an address. The screenshot above shows that the latitude/longitude option has been selected.
- *Location:* The geolocation to be shown in the map. If a latitude/longitude coordinate is indicated in the previous *Specify Location As* setting, then the coordinates of the location must be entered as an XPath expression (as shown in the screenshot above). The generated string must have one of the lexical formats described in the *Geolocation input string formats* section below. If the *Address* option is selected, then the *Location* entry could be something like: `"Rudolfsplatz 13a, 1010 Vienna"`.
- *Label:* The text of the label that is attached to the geolocation shown in the map. The text is entered as an XPath expression that generates an `xs:string`.
- *Zoom:* The zoom factor of the map when it is opened in the map application.

## Show route from current location

The *Show route from current location* action opens the map application of the mobile device, and shows the route from the current location to the specified geolocation.



OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
○ Show Geolocation ● Show route from current location ○ Show route between locations
Specify Locations as ● Latitude/Longitude ○ Address
To Location "0°N 0°E"
Label "Greenwich at Equator"
Routing type Drive

The *Show route from current location* action has the following settings:

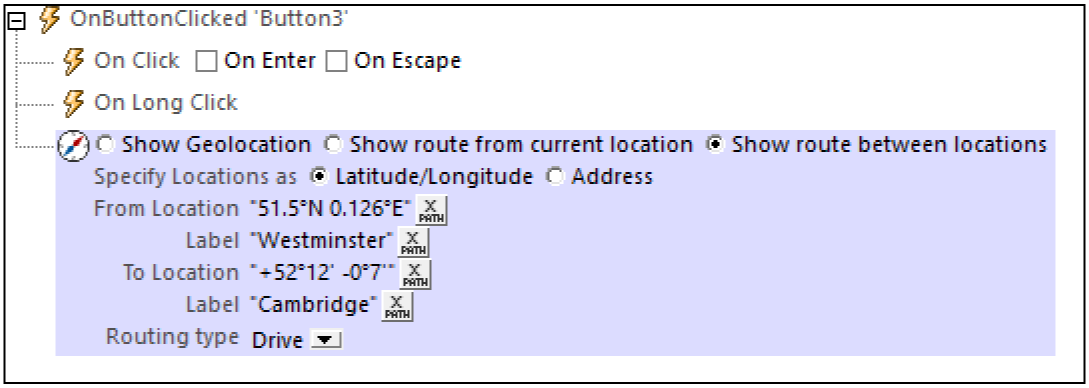
- *Specify Location as:* Whether the entry of the *Location* setting (next setting) is a latitude/longitude coordinate or an address. The screenshot above shows that the latitude/longitude option has been selected.
- *To Location:* The destination location to be shown in the map. If a latitude/longitude coordinate is indicated in the previous *Specify Location As* setting, then the coordinates of the location must be entered as an XPath expression (as shown in the screenshot above). The generated string must have

one of the lexical formats described in the *Geolocation input string formats* section below. If the *Address* option is selected, then the *Location* entry could be something like: `"Rudolfsplatz 13a, 1010 Vienna"`.

- *Label:* The text of the label that is attached to the destination location shown in the map. The text is entered as an XPath expression that generates an `xs:string`.
- *Routing type:* Select the type of transport that will be used to travel the route: (i) *Drive* (private transport), (ii) *Walk*, (iii) *Public Transport*.

## Show route between locations

The *Show route between locations* action opens the map application of the mobile device, and shows the route between the two specified geolocations.



OnButtonClicked 'Button3'

On Click □ On Enter □ On Escape

On Long Click

○ Show Geolocation ○ Show route from current location ● Show route between locations

Specify Locations as ● Latitude/Longitude ○ Address

From Location "51.5°N 0.126°E"

Label "Westminster"

To Location "+52°12' -0°7'"

Label "Cambridge"

Routing type Drive

The *Show route between locations* action has the following settings:

- *Specify Location as:* Whether the entry of the *Location* settings are latitude/longitude coordinates or addresses. The screenshot above shows that the latitude/longitude option has been selected.
- *From/To Location:* The start and destination locations, respectively, to be shown in the map. The coordinates of the locations are entered as XPath expressions. If latitude/longitude coordinates are indicated in the previous *Specify Location As* setting, then the coordinates of the locations must be entered as XPath expressions (as shown in the screenshot above). The generated strings must have one of the lexical formats described in the *Geolocation input string formats* section below. If the *Address* option is selected, then the *Location* entries could be something like: `"Rudolfsplatz 13a, 1010 Vienna"`.
- *Label:* The text of the label that is attached to the start and destination locations, respectively. Each text is entered as an XPath expression that generates an `xs:string`.
- *Routing type:* Select the type of transport that will be used to travel the route: (i) *Drive* (private transport), (ii) *Walk*, (iii) *Public Transport*.

⊟ *Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (`"`) while unit indicators that are escaped are highlighted in blue (`""`).

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example*: **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example*: **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example*: **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example*: **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example*: **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example*: **`33.33  -22.22`**

*Examples of format-combinations:*

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

For information about specifying geolocation data for designer and server simulations, see the section Geolocation Settings[1372].

## MobileTogether extension functions

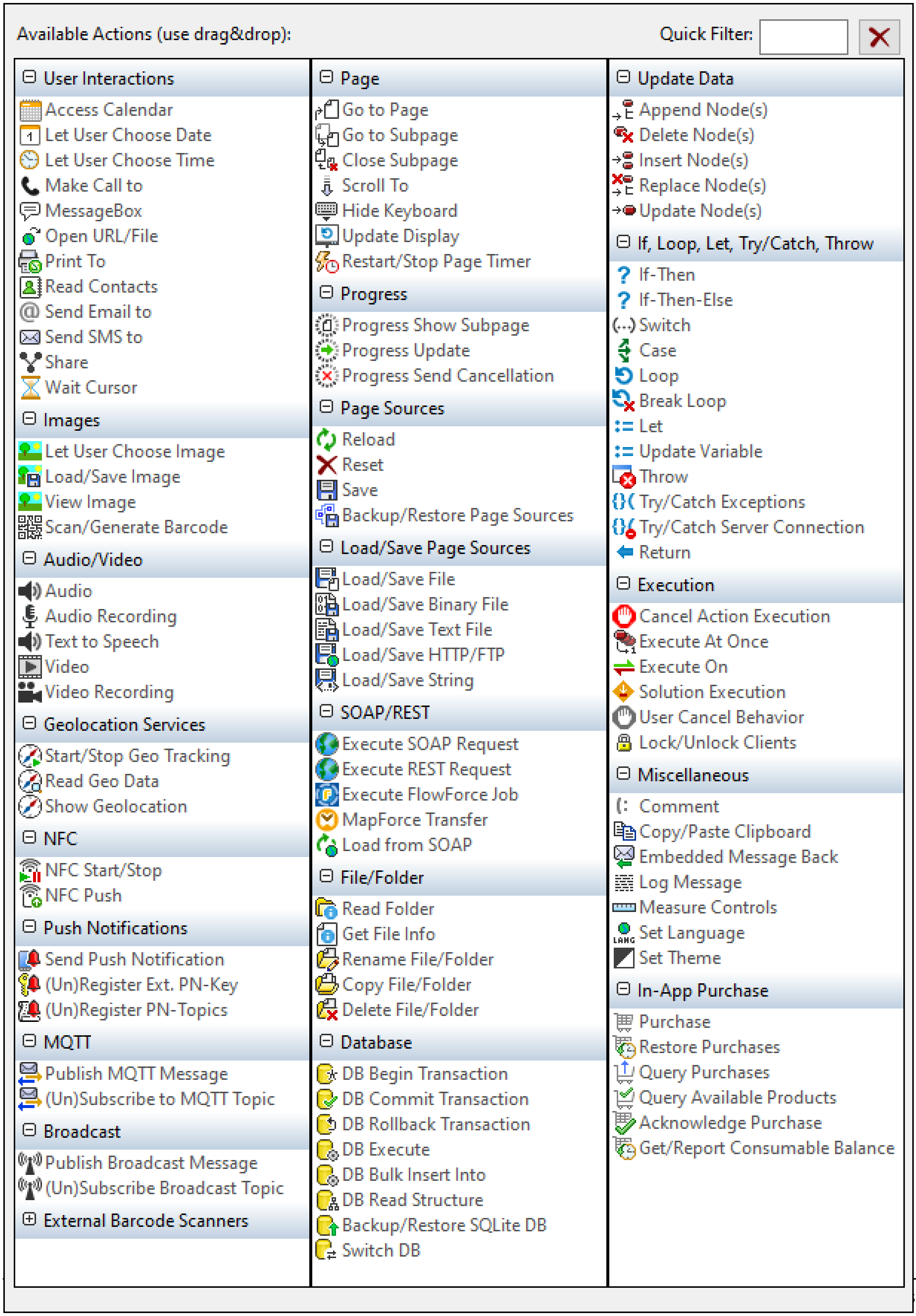
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-geo-map-marker()`**
**`mt-geolocation-started()`**

# 10.5 NFC

The following actions are available in the NFC group of the Actions dialog (*screenshot below*):

- NFC Start/Stop 746
- NFC Push



Available Actions (use drag&drop): Quick Filter:

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

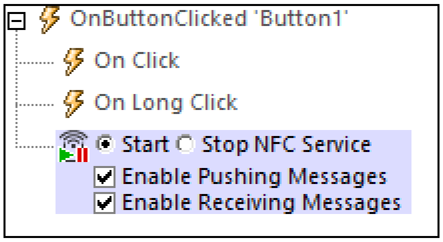
⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events 389 and Control Events 665*.

See the section NFC 1118 for a broader description of NFC support in MobileTogether.

## 10.5.1 NFC Start/Stop

The NFC Start/Stop action (*screenshots below*) is used to start or stop the pushing and/or receiving of messages.



OnButtonClicked 'Button1'
- On Click
- On Long Click
- ● Start ○ Stop NFC Service
  - ☑ Enable Pushing Messages
  - ☑ Enable Receiving Messages

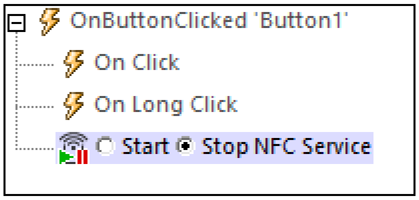
Select the *Start* options you want. After the action has been added to the design, a `$MT_NFC` tree is automatically added to the Page Sources Pane 270 (*see tree structure below*).

```
<Root>
   <Tag Id=""/>
   <NdefMessage
      CanMakeReadOnly=""
      IsWriteable=""
      MaxSize=""
      Type="">
      <NdefRecord
         Id=""
         TypeNameField=""
         RecordTypeDefinition=""
         Type=""
         Text=""
         Language=""
         URI=""
         Payload=""
         MimeType=""
         ExternalDomain=""
         ExternalPackageName="">
         <NdefRecord />
      </NdefRecord>
      <NdefRecord />
         ...
      <NdefRecord />
   </NdefMessage>
</Root>
```

Pushing and/or receiving is started when the ***Start*** action is triggered. The sequence of steps that the action sets in motion is as follows:

1. NFC must be enabled on the device. If NFC is not enabled, then triggering the *Start* action will cause a prompt to appear asking the user to enable NFC.
2. After ensuring that NFC is enabled, the MobileTogether Client app will be registered for NFC.
3. Immediately thereafter, NFC tag discovery is automatically started and NFC messages in NFC tags will be automatically received. Pushing can be started via an NFC Push action 747; it is not automatically started.
4. Received data is stored in the `$MT_NFC` tree.

The ***Stop*** action stops the pushing and receiving of all messages.



To re-start the pushing and receiving of messages, trigger the *Start* action again.

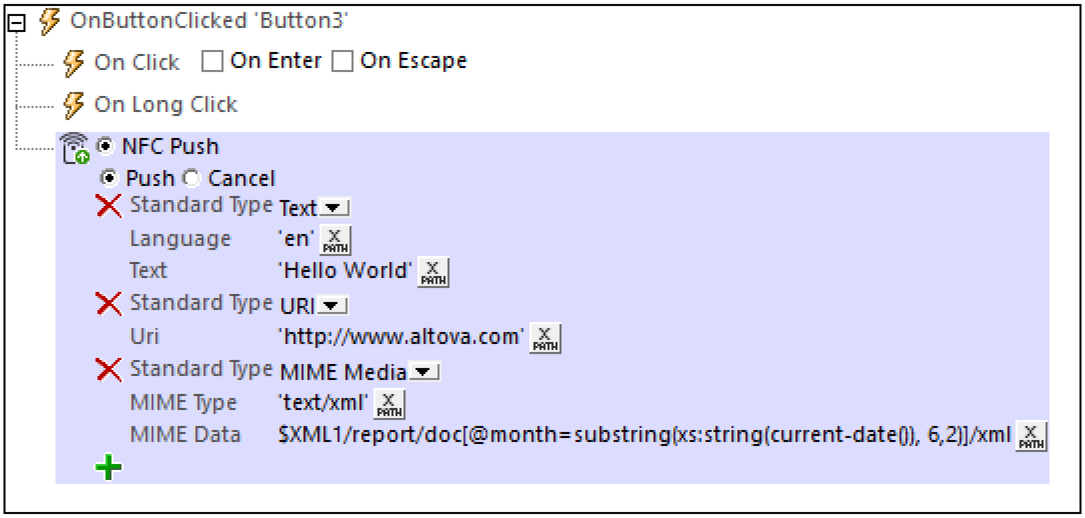
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-nfc-started()`**

## 10.5.2 NFC Push

The NFC Push action (*screenshot below*) defines the message or file to push. When the action is triggered, the specified message or file is transmitted via NFC. In order for the Push action to work, the NFC Start action 746 must be triggered already—so that the device is ready for the Push action. Once an NFC Push action has been started, the NFC message or file will be transmitted to any receiving device that is placed within NFC range of the sending device. This continuous sending is stopped (i) when NFC is stopped 746 or (ii) when an NFC Push 747 action is canceled (which is done by adding a new NFC Push action with the *Cancel* option selected; *see the action's screenshot below*).



OnButtonClicked 'Button3'

On Click □ On Enter □ On Escape

On Long Click

● NFC Push

● Push ○ Cancel

Standard Type Text

Language 'en'

Text 'Hello World'

Standard Type URI

Uri 'http://www.altova.com'

Standard Type MIME Media

MIME Type 'text/xml'

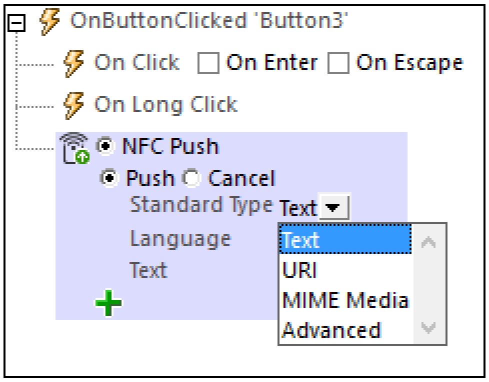
MIME Data $XML1/report/doc[@month=substring(xs:string(current-date()), 6,2)]/xml

Each NFC Push action can specify either:

- *Push:* To start the pushing of a message. Each NFC Push action corresponds to one message to be transmitted. Note, however, that a message can consist of multiple records. For example, the message in the screenshot above contains three records. If you want to send more than one message, then add a new NFC Push action for each message.
- *Cancel:* To cancel the the pushing of a message. Add the NFC Push action and use the action's *Cancel* option to stop any message pushing that has been previously triggered by some other NFC Push action..

## Pushing a message

The NFC Push option enables you to define the NFC message that you want to transmit. In the combo box (*see screenshot below*), select the Standard Type of the message to be transmitted. These types are as defined in the NFC technical specifications.



OnButtonClicked 'Button3'

On Click □ On Enter □ On Escape

On Long Click

● NFC Push

● Push ○ Cancel

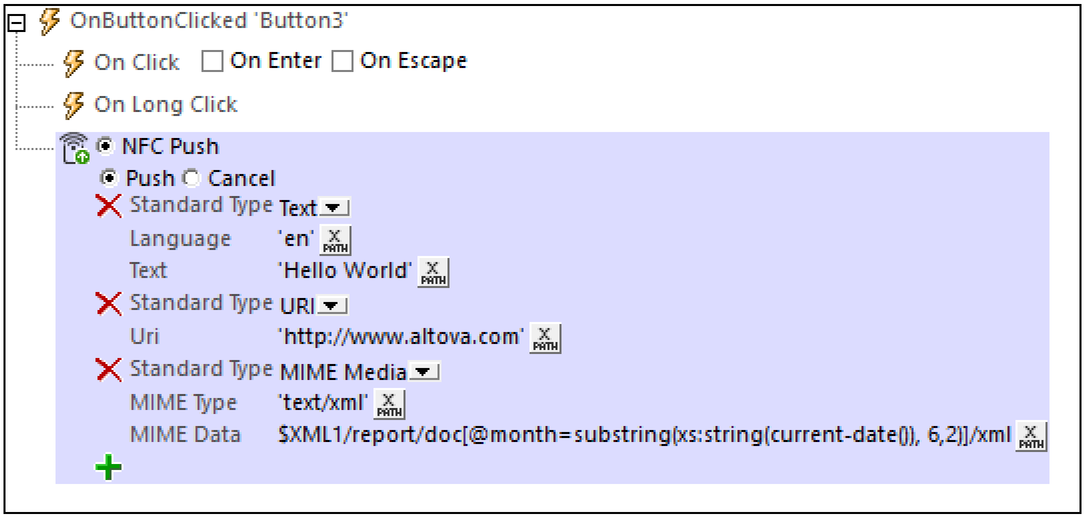
Standard Type Text

Language

Text

*Text, URI, and MIME-Media types*

Definitions of the `Text`, `URI`, and `MIME Media` standard types are straightforward (but see the URI options below the next screenshot). The screenshot below shows an NFC message with three records, each of which defines a different standard type. Message contents can be XPath strings, or can come from source tree nodes. In the last record, for example, the message's contents are taken from a node in an XML page source 347.



⊟ OnButtonClicked 'Button3'

On Click □ On Enter □ On Escape

On Long Click

● NFC Push

● Push ○ Cancel

Standard Type Text

Language 'en'

Text 'Hello World'

Standard Type URI

Uri 'http://www.altova.com'

Standard Type MIME Media

MIME Type 'text/xml'

MIME Data $XML1/report/doc[@month=substring(xs:string(current-date()), 6,2)]/xml

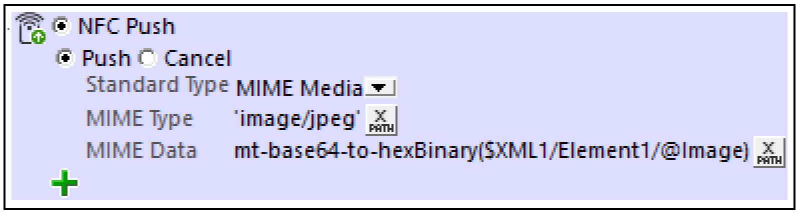
The following URI options are available:

- *SMS URI:* The sent SMS will be opened in the receiving device's SMS app. The URI would look something like this: **`"sms:+439991234567?body=MyBody"`**
- *Telephone URI:* The telephone app on the receiving device will be opened, and the transmitted telephone number will be dialed. Example URI: **`"tel:+439991234567"`**
- *Email URI:* The sent email will be opened in the receiving device's email app. Example URI: **`"mailto:name@altova.com?subject=MySubject&body=MyBody"`**

In some types of messages (*see the NFC technical specifications*), the message content must be explicitly converted to hexBinary so that it can be stored and transported in the payload of the NFC message[1118]. If you wish to convert a string to hexBinary, you can use the XPath extension function[1262] `mt-string-to-hexBinary`. Images can be converted from Base64[1098] to hexBinary with the `mt-base64-to-hexBinary` function[1262]. When an NFC payload is received on a device (*see Discovering and Reading NFC Tags*[1119]), the hexBinary can be converted back to a string or a Base64 image by using, respectively, the XPath extension functions[1262] `mt-hexBinary-to-string` and `mt-hexBinary-to-base64`.

*Pushing images*

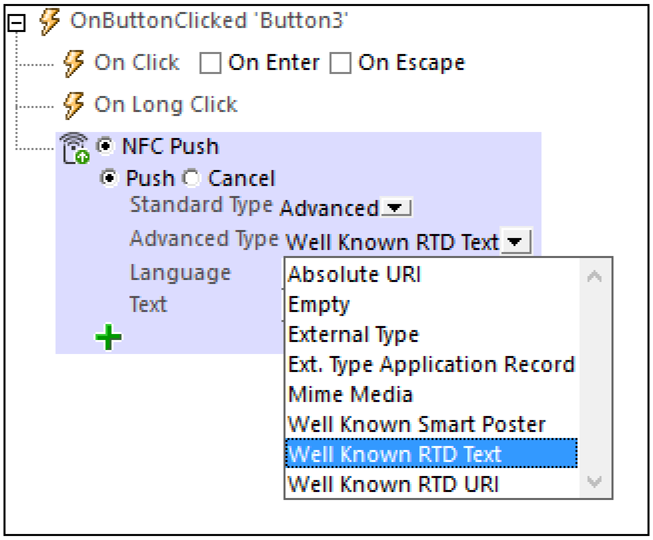
If you wish to transmit an image via an NFC Push message, you can do this by using the relevant MIME media type for that image (*see screenshot below for an example*). When defining this kind of message, you will need to convert the Base64 format of the image[1098] to hexBinary so that the image can be transported in the payload of the NFC message[1118]. For this conversion, use the XPath extension function[1262] `mt-base64-to-hexBinary`. To convert the hexBinary (on the receiving device) back to a Base64 image, use the function[1262] `mt-hexBinary-to-base64`.



● NFC Push
● Push ○ Cancel
Standard Type MIME Media
MIME Type 'image/jpeg'
MIME Data mt-base64-to-hexBinary($XML1/Element1/@Image)

*Advanced types*

For messages that are of the `Advanced` type, a specific advanced type must be selected (*see screenshot below*). For each advanced type, enter the respective text and specify any other relevant information.



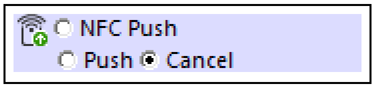
OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
● NFC Push
● Push ○ Cancel
Standard Type Advanced
Advanced Type Well Known RTD Text
Language
Text

- Absolute URI
- Empty
- External Type
- Ext. Type Application Record
- Mime Media
- Well Known Smart Poster
- Well Known RTD Text
- Well Known RTD URI

For a description of NFC's advanced types, see the NFC technical specifications.

## Canceling a push/beam

The Cancel action enables you to cancel a previously triggered push action that has not been stopped.



○ NFC Push
○ Push ● Cancel

## MobileTogether extension functions

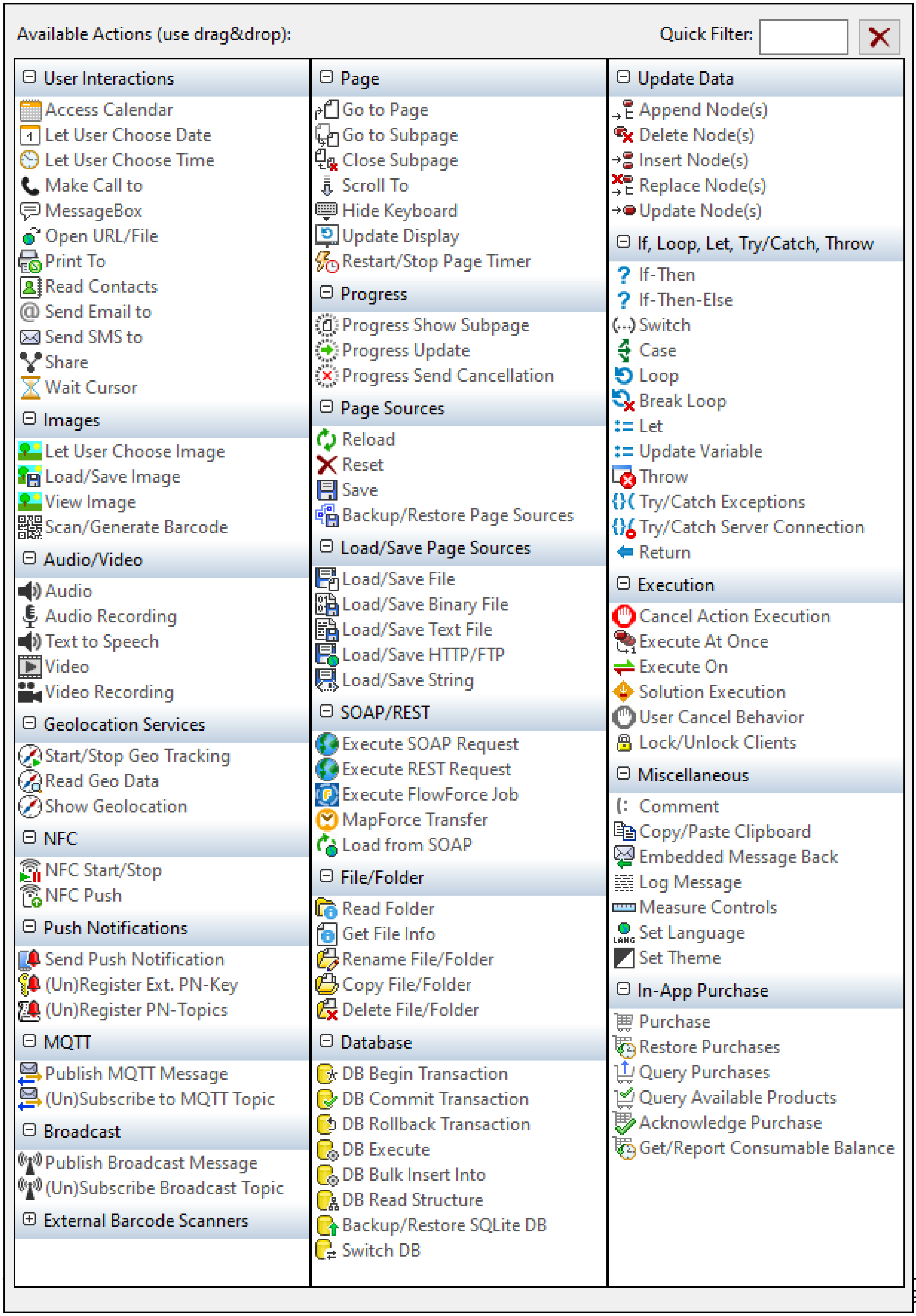
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-nfc-started()`**

## 10.6 Push Notifications

The following actions are available in the Push Notifications group of the Actions dialog (*screenshot below*):

- Send Push Notification 753
- (Un)Register Ext PN-Key 757
- (Un)Register PN-Topics



Available Actions (use drag&drop): Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

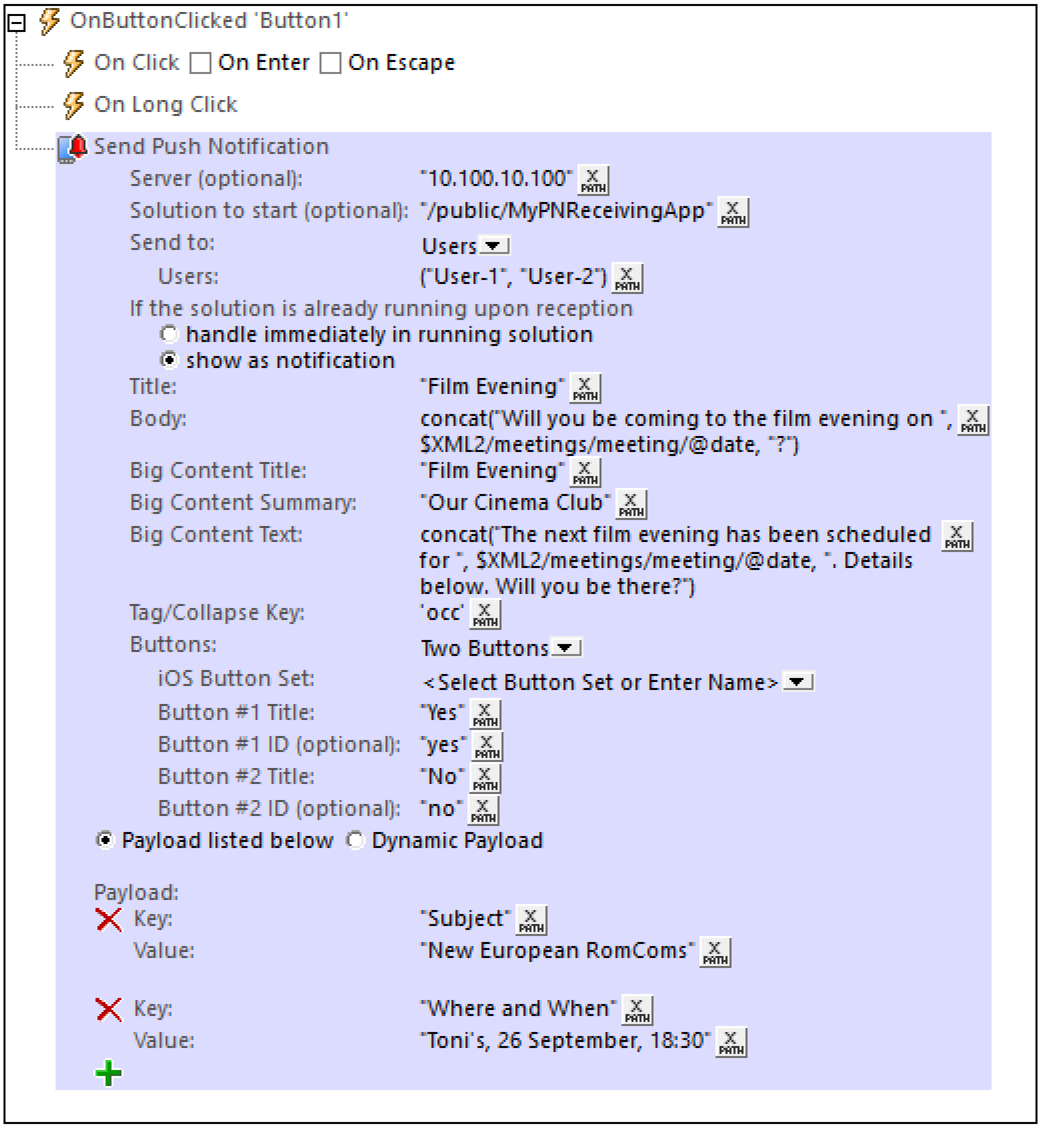
⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events* 389 *and* *Control Events* 665.

See the section Push Notifications 1125 for an overview of how to use push notifications in MobileTogether.

## 10.6.1 Send Push Notification

When the Send Push Notification action (*screenshot below*) is triggered, a push notification (PN) is sent according to the action's settings (*described below*). The PN is sent to a receiving solution, which can be: (i) the same solution hosted on the same server, or (ii) another solution that is hosted either on the same server (as the sending solution) or on another server.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Send Push Notification
Server (optional): "10.100.10.100"
Solution to start (optional): "/public/MyPNReceivingApp"
Send to: Users
Users: ("User-1", "User-2")
If the solution is already running upon reception
○ handle immediately in running solution
● show as notification
Title: "Film Evening"
Body: concat("Will you be coming to the film evening on ", $XML2/meetings/meeting/@date, "?")
Big Content Title: "Film Evening"
Big Content Summary: "Our Cinema Club"
Big Content Text: concat("The next film evening has been scheduled for ", $XML2/meetings/meeting/@date, ". Details below. Will you be there?")
Tag/Collapse Key: 'occ'
Buttons: Two Buttons
iOS Button Set: <Select Button Set or Enter Name>
Button #1 Title: "Yes"
Button #1 ID (optional): "yes"
Button #2 Title: "No"
Button #2 ID (optional): "no"
● Payload listed below ○ Dynamic Payload

Payload:
Key: "Subject"
Value: "New European RomComs"

Key: "Where and When"
Value: "Toni's, 26 September, 18:30"

The action's settings, which are listed below, define the different parameters of the push notification:

⊟ *For a MobileTogether Server Service, select Solution or AppStore App*

If the Send Push Notification action is being created in a MobileTogether Server Service[1543], then you must specify whether the PN is to be sent to (i) a MobileTogether solution, or (ii) a MobileTogether AppStore App[1471]. Select the appropriate radio button to do this. Note that this option is displayed only if the Send Push Notification action is created in a server service; it is not displayed in designs for standard solutions or AppStore Apps.

⊟ *Server (optional)*

If this option is not specified, then the PN is assumed to be targeted at a receiving solution that is being hosted on the same server as the sending solution. If the receiving solution is on a different server than the sending solution, then this server is specified here. The setting's XPath expression must resolve to a string that is the IP address of the relevant server (*see screenshot above*).

⊟ *Solution to start (optional)*

If this option is not specified, then, on the receiving device, the same solution as the sending solution is started when the PN is received. If the receiving solution is different than the sending solution, then the receiving solution is specified in this setting. The setting's XPath expression must resolve to a string that is the path to the solution's workflow on the server. In the screenshot above, for example, the receiving solution is `MyPNReceivingApp`, which is located in the `public` container on the server with the IP address `10.100.10.100`.

⊟ *Send to*

Specifies the recipients of the PN. The following options are available:

- *Users:* The PN will be sent to user/s on the receiving-solution's server that are listed in the *Users* setting. The *Users* setting takes as its input a single string item (for example: `'User-1'`) or a sequence of string items (for example: `('User-1', 'User-2')`).
- *All users of solution:* All users that have permission to access the receiving solution (on the receiving-solution's server). (User permissions to access a workflow are defined in the administrator settings of MobileTogether Server.)
- *Roles:* The PN will be sent to users that have the role/s specified in the *Roles* setting The *Roles* setting takes as its input a single string item (for example: `'Role-1'`) or a sequence of string items (for example: `('Role-1', 'Role-2')`). (For more information about roles, see the MobileTogether Server documentation.)
- *External PN Keys:* The PN will be sent to devices that have been registered with any one of the external PN keys specified in the *External Keys* setting. The *External Keys* setting takes as its input a single string item (for example: `'Key-1'`) or a sequence of string items (for example: `('Key-1', 'Key-2')`). See Register/Unregister Ext PN-Key[757] for more information.
- *Topics:* The PN will be sent to devices that have subscribed to any one of the topics specified in the *Topics* setting. The *Topics* setting takes as its input a single string item (for example: `'Topic-1'`) or a sequence of string items (for example: `('Topic-1', 'Topic-2')`). See Register/Unregister PN-Topics[759] for more information on PN topics.

⊟ *If the solution is already running when the PN is received*

Two possibilities exist:

- The actions that are defined for the **`OnPushNotificationReceived`** 296 event of the receiving solution can be executed immediately. This is done silently; no PN is displayed.
- The PN is displayed. The actions that are defined for the **`OnPushNotificationReceived`** 296 event of the receiving solution are executed when the solution's user taps the PN—or an appropriate button in the PN.

*Title, Body*

A PN consists of a short message, which, if tapped, opens the big message. The *Title* and *Body* settings are the text strings, respectively, of the title and text of the short message. If buttons have been specified for the PN (*see farther below*), then these may, depending on the OS, appear below the short message.

*Title/Summary/Text of Big Content*

A PN consists of a short message, which, if tapped, opens the big message. The Big Content Title, Summary, and Text settings define the text strings, respectively, of the big message's title, summary, and body. The big message is optional, and these settings can be left empty. If buttons have been specified for the PN (*see farther below*), then these appear in the big message.

**Note:** The Big Content of standard MobileTogether solutions is displayed only on Android and Windows devices. If you want Big Content to be displayed on iOS devices, then compile the receiving solution as an AppStore App 1130.

*Tag/Collapse Key*

A tag/collapse key is a text string that is sent with a PN. It is generated by the setting's XPath expression, which must resolve to a string. If the tag/collapse key is specified, then (i) the PNs generated by this action will have the key that is generated by the XPath expression of this option, and (ii) all PNs with this key on the receiving device will be collapsed to the last PN that is received. If the tag/collapse key is not specified, then all PNs sent by this action will be displayed. Note also that, if multiple Send Push Notification actions generate the same tag/collapse key, then the PNs sent by these different actions will all have the same key and will all be collapsed.

*PN Buttons*

Specifies the number of buttons to show in the PN, from none to three. The purpose of buttons in the PN is to enable the user to select one set of actions from among one to three sets of actions (one set of actions is defined per button). In the case of non-iOS mobile devices, specify the title and optional ID of individual PN buttons. In the case of iOS devices (on which PN buttons are available for AppStore Apps 1471 only), select a PN *button set* to use.

- *iOS Button Set:* Buttons for PNs that appear on iOS devices can be displayed in AppStore Apps, not in standard MobileTogether solutions. Enter or select the name of the PN button set that you want to display. The buttons of this PN button set will be shown in the PN. The PN button set that is specified here must be defined in the receiving solution (via the **iOS Push Notification Button Sets** 1608 command). If the receiving solution is the same as the sending solution, then the PN button sets that have been defined are shown in the option's combo box. For example, if the

*Buttons* option has been set to `Two Buttons`, then the available PN button sets for two buttons are displayed in the combo box. If the receiving solution is not the same as the sending solution, then the name of the PN button set must be entered in this setting.

- *Individual PN buttons for non-iOS devices:* The *Title* setting is the text that is displayed on the PN button. The *ID* setting takes the PN button number as its default value. For example Button #1 has an id of `"1"`. You can optionally enter any ID that you like. The value of the *ID* setting is used in the receiving solution to reference the user's button selection.

**Note:** PN buttons are not related in any way to Button controls[417].

In the receiving solution, when the user taps a PN button, that button's ID is **automatically** entered as the value of the `$MT_PUSHNOTIFICATION/Root/@button` node (*see structure of the page source below*). So if a user taps Button #1, then the `$MT_PUSHNOTIFICATION/Root/@button` node will contain the ID of Button #1. Using conditional statements that test the value in this node, a set of actions[667] can be carried out according to which PN button the user has tapped. These conditional actions to carry out are defined in the `OnPushNotificationReceived`[296] event of the receiving solution.

The structure of the `$MT_PUSHNOTIFICATION` page source:

```
$MT_PUSHNOTIFICATION
Root
|   @button
|
|-- Entry
|      @key
|      @value
```

⊟ *Payload*

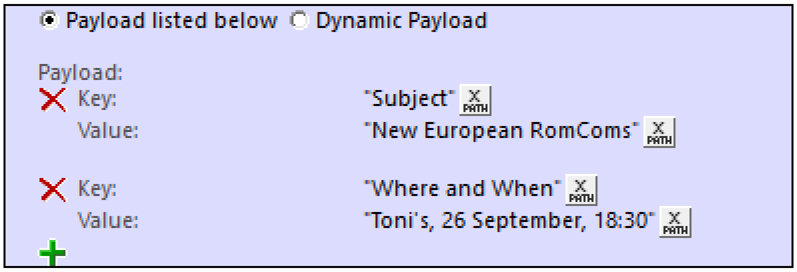
The payload is the data that is sent with the PN and saved to the `$MT_PUSHNOTIFICATION` page source of the receiving solution (*see its structure below*). The data in this page source can then be used in the design of the receiving solution.

The payload is sent as an array of key–value pairs. Each key–value pair is placed (in index order) in an **`Entry`** element of the receiving solution's `$MT_PUSHNOTIFICATION` page source.

```
$MT_PUSHNOTIFICATION
Root
|   @button
|
|-- Entry
|      @key
|      @value
```

You can specify the payload of the PN in the following ways:

- As a list of key–value pairs (*see screenshot below*). Each pair in the list is added by clicking the plus symbol, then entering an XPath expression, respectively, for the key and its value. Each XPath expression must evaluate to a single string. The string-value (or name) of each *key* must be non-empty and unique. The string-value of a *value* need not be unique and can be empty.

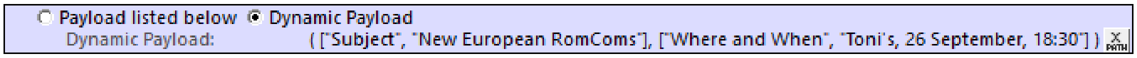


● Payload listed below ○ Dynamic Payload

Payload:

| | |
|---|---|
| Key: | "Subject" |
| Value: | "New European RomComs" |
| Key: | "Where and When" |
| Value: | "Toni's, 26 September, 18:30" |

- As a dynamically determined array of key–value pairs. The screenshot below shows how such an array would look if entered directly as an XPath expression. The array could, however, be obtained dynamically at run time by iterating over a set of nodes. Note that the string-value of each key must be non-empty and unique.



○ Payload listed below ● Dynamic Payload

Dynamic Payload: ( ["Subject", "New European RomComs"], ["Where and When", "Toni's, 26 September, 18:30"] )

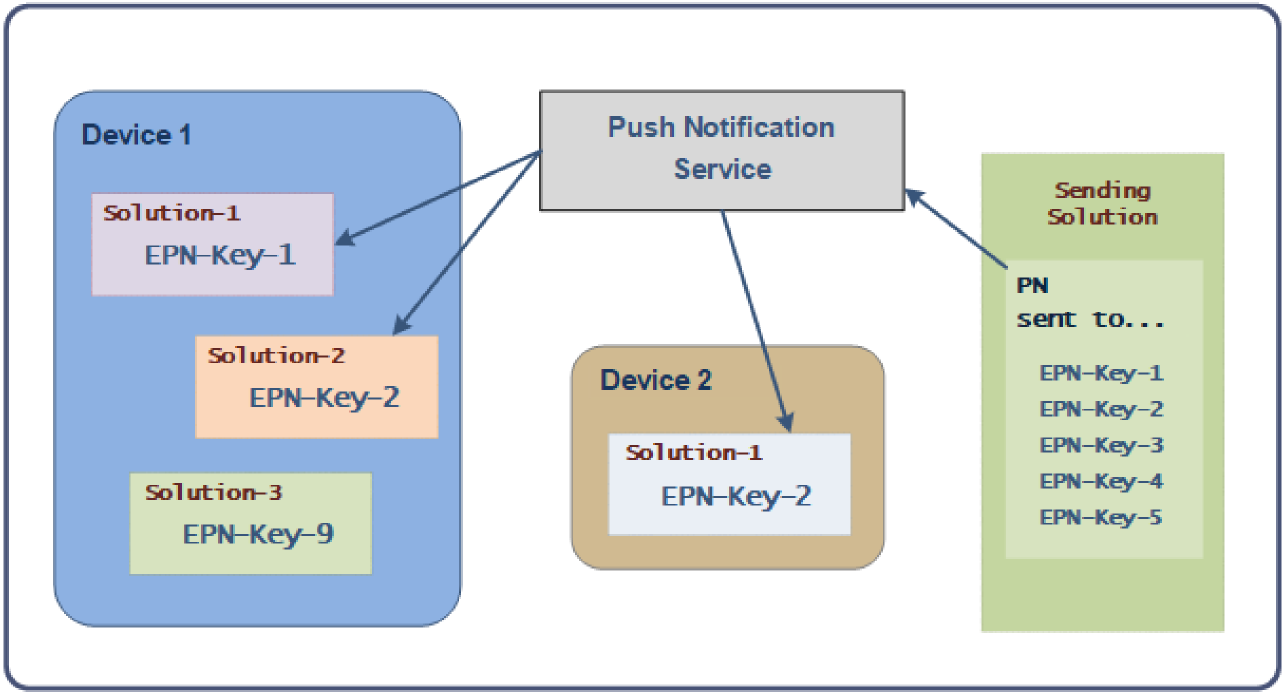
**Note:** The names of payload keys must not start with `mt_` and must not be any of the parameter names documented here: https://firebase.google.com/docs/cloud-messaging/http-server-ref.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.6.2 (Un)Register Ext PN-Key

This action is used in a solution on a PN-receiving mobile device. It registers a text string as the external PN key of *that solution on that mobile device*. Note that the device is registered with the specified external PN key for a particular solution. Multiple solutions on a device can each register a different external PN key.



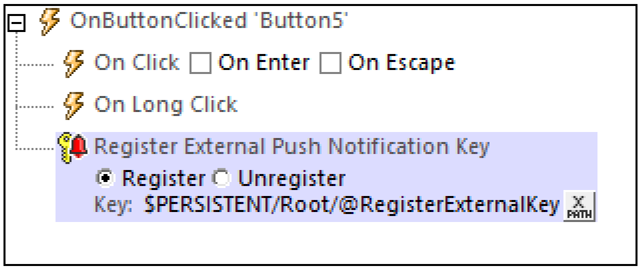
When sending a PN[753], there are different ways to define a set of PN receivers. One way is to specify that the PN is sent to one or more external PN keys (*see Send Push Notification*[753]). So, if a device has been registered using one of the targeted external PN keys, then the PN will be sent to that device—more specifically, to the receiving solution on that device that registered the target PN key.

**Note:** External PN keys can also be used in AppStore Apps[1471].

*See also: The Receiving Solution*[1128]

## Registering

To register an external PN key, select *Register*, and specify a key via an XPath expression. The expression must resolve to a text string. In the screenshot below, for example, when a button is clicked, a string located in a page source node is registered as the external PN key of that device. Now, if a PN is sent to this key, then that mobile device will receive the PN.



OnButtonClicked 'Button5'

On Click □ On Enter □ On Escape

On Long Click

Register External Push Notification Key

● Register ○ Unregister

Key: $PERSISTENT/Root/@RegisterExternalKey

### Unregistering

If at some point the user wishes to not receive PNs that are addressed to a given external PN key, then the *Unregister* action can be used to cancel the registration. This could be done, for example, on a button tap.

To unregister a PN, add the action and select *Unregister*. Since only one external PN key exists for a solution at any given time, you do not need to specify the key.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.6.3 (Un)Register PN-Topics

This action is used in a solution on a PN-receiving mobile device for registering that device to receive PNs about one or more selected topics. For example, if a device is registered to receive PNs about topics marked as **`news`** and topics marked as **`travel`**, then it will receive any PN that is marked as either **`news`** or **`travel`**. A PN is marked with a topic when it is sent (*see Send Push Notification* 753). For example, if a PN is sent to the topics **`news`** and **`travel`**, it will be received by any device that is registered for either (or both) of these topics.

Note that it is the device that is registered for topics, so the action can be triggered from any solution on that device. It would, however, be logical and more intuitive to use the PN-receiving solution to register the PN topic.

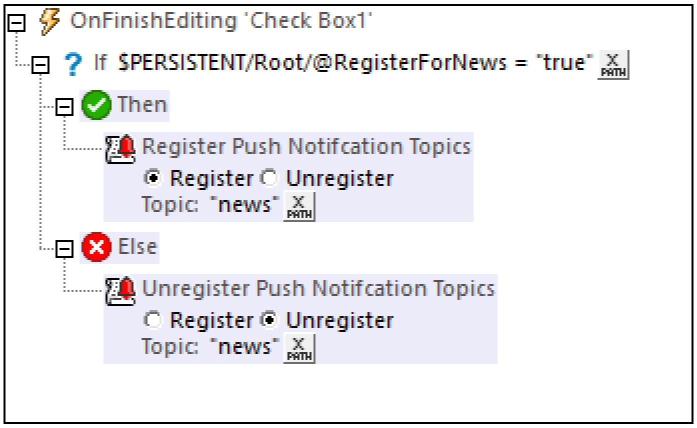
Among different ways to define a set of PN receivers when sending a PN 753, you can specify that the PN be sent to one or more specific topics (*see Send Push Notification* 753). If a device is registered for any one of these topics, then the device will receive that PN.

**Note:** PN topics are supported on Android and iOS; they are **not supported on Windows**.

*See also: The Receiving Solution* 1128

### Registering

To register an external PN topic, select *Register*, and specify a topic via an XPath expression. The expression must resolve to a string (for example: `'news'`) or a sequence of strings (for example: `('news', 'travel')`). In the screenshot below, for example, Register/Unregister PN Topics actions are defined for a check box which is linked to a page source node. As a result of this linkage, the page source node will have a value of **`true`** (when the check box is checked) or **`false`** (when it is unchecked). Depending on whether the user checks the box or not, the device will be registered for the **`news`** topic or not (*see screenshot*) and accordingly will receive or will not receive PNs that are sent to the **`news`** topic.



```
OnFinishEditing 'Check Box1'
  If $PERSISTENT/Root/@RegisterForNews = "true"
    Then
      Register Push Notifcation Topics
        ● Register ○ Unregister
        Topic: "news"
    Else
      Unregister Push Notifcation Topics
        ○ Register ● Unregister
        Topic: "news"
```

## Unregistering

If at some point the user wishes to not receive PNs that are sent to a topic for which the device is registered, then the *Unregister* action can be used to cancel the registration. This could be done, for example, on a button tap.

To unregister a PN, add the action, select *Unregister*, and enter the topic for which the device is to be unregistered. To unregister multiple topics, enter the topics as a sequence of strings (for example: `('news', 'travel')`).

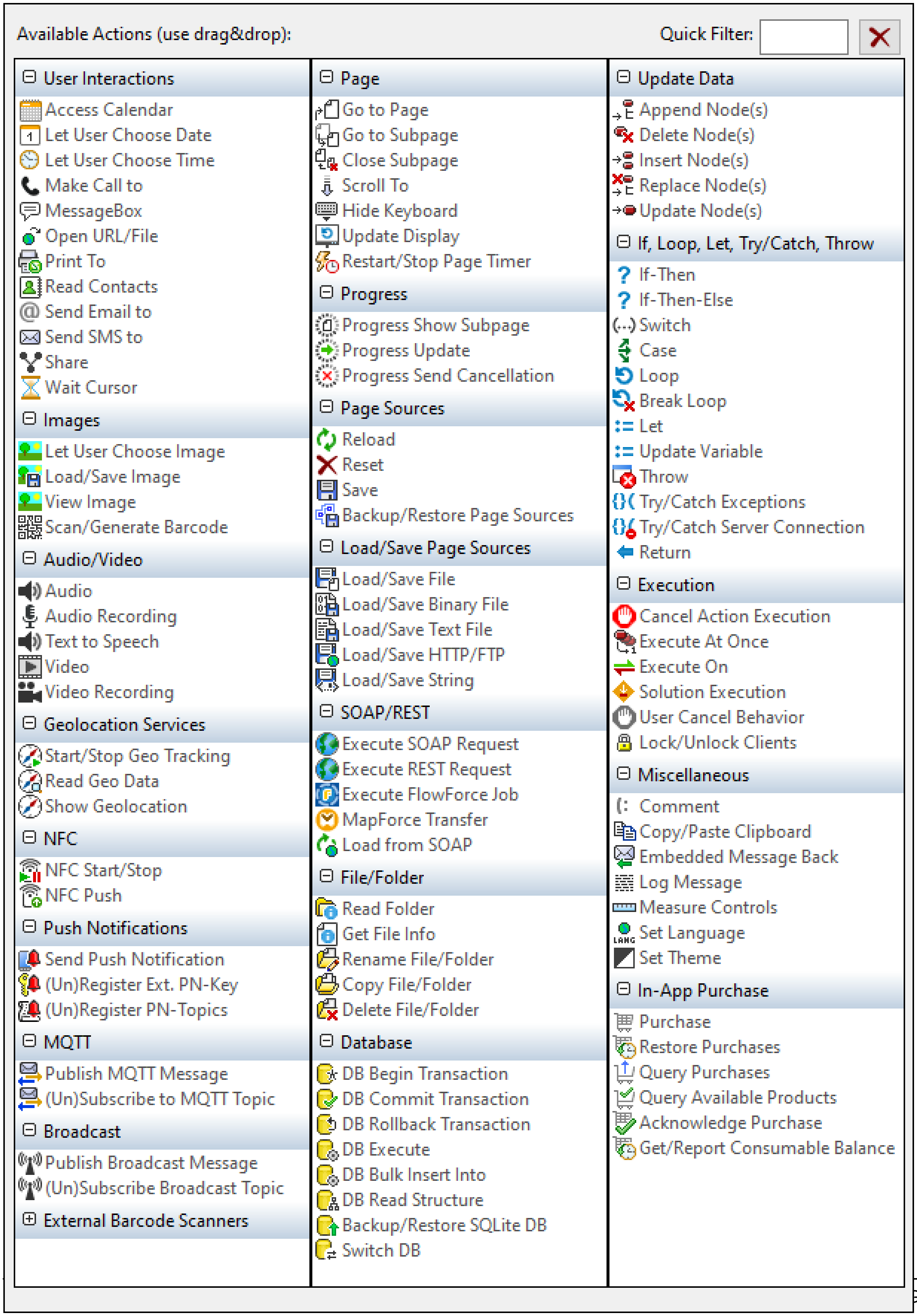
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

# 10.7 MQTT

The following actions are available in the MQTT group of the Actions dialog (*screenshot below*):

- Publish MQTT Message 763
- (Un)Subscribe to MQTT Topic

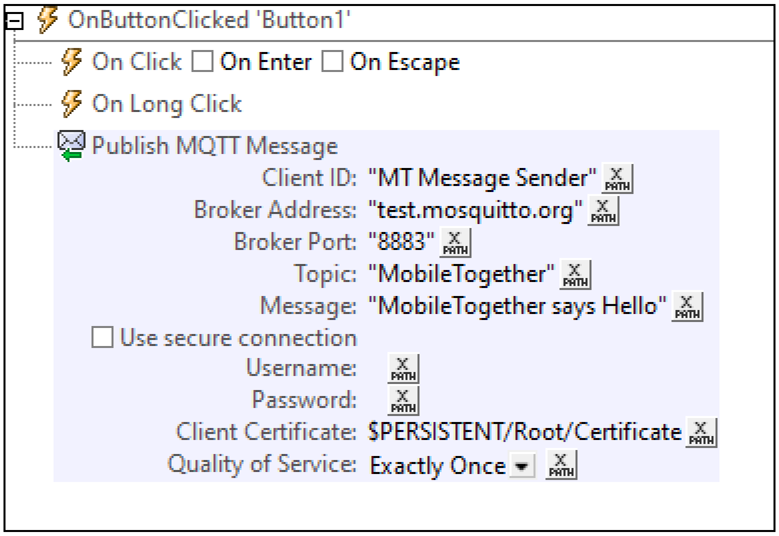

Available Actions (use drag&drop):
Quick Filter:
⊟ User Interactions
Access Calendar
Let User Choose Date
Let User Choose Time
Make Call to
MessageBox
Open URL/File
Print To
Read Contacts
Send Email to
Send SMS to
Share
Wait Cursor
⊟ Images
Let User Choose Image
Load/Save Image
View Image
Scan/Generate Barcode
⊟ Audio/Video
Audio
Audio Recording
Text to Speech
Video
Video Recording
⊟ Geolocation Services
Start/Stop Geo Tracking
Read Geo Data
Show Geolocation
⊟ NFC
NFC Start/Stop
NFC Push
⊟ Push Notifications
Send Push Notification
(Un)Register Ext. PN-Key
(Un)Register PN-Topics
⊟ MQTT
Publish MQTT Message
(Un)Subscribe to MQTT Topic
⊟ Broadcast
Publish Broadcast Message
(Un)Subscribe Broadcast Topic
⊞ External Barcode Scanners
⊟ Page
Go to Page
Go to Subpage
Close Subpage
Scroll To
Hide Keyboard
Update Display
Restart/Stop Page Timer
⊟ Progress
Progress Show Subpage
Progress Update
Progress Send Cancellation
⊟ Page Sources
Reload
Reset
Save
Backup/Restore Page Sources
⊟ Load/Save Page Sources
Load/Save File
Load/Save Binary File
Load/Save Text File
Load/Save HTTP/FTP
Load/Save String
⊟ SOAP/REST
Execute SOAP Request
Execute REST Request
Execute FlowForce Job
MapForce Transfer
Load from SOAP
⊟ File/Folder
Read Folder
Get File Info
Rename File/Folder
Copy File/Folder
Delete File/Folder
⊟ Database
DB Begin Transaction
DB Commit Transaction
DB Rollback Transaction
DB Execute
DB Bulk Insert Into
DB Read Structure
Backup/Restore SQLite DB
Switch DB
⊟ Update Data
Append Node(s)
Delete Node(s)
Insert Node(s)
Replace Node(s)
Update Node(s)
⊟ If, Loop, Let, Try/Catch, Throw
If-Then
If-Then-Else
Switch
Case
Loop
Break Loop
Let
Update Variable
Throw
Try/Catch Exceptions
Try/Catch Server Connection
Return
⊟ Execution
Cancel Action Execution
Execute At Once
Execute On
Solution Execution
User Cancel Behavior
Lock/Unlock Clients
⊟ Miscellaneous
Comment
Copy/Paste Clipboard
Embedded Message Back
Log Message
Measure Controls
Set Language
Set Theme
⊟ In-App Purchase
Purchase
Restore Purchases
Query Purchases
Query Available Products
Acknowledge Purchase
Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *[Page Events](389)* *and* *[Control Events](665)*.

See the section [MQTT](1135) for an overview of how you can use MobileTogether solutions with MQTT.

## 10.7.1 Publish MQTT Message

The Publish MQTT Message action (*screenshot below*) publishes a message to the specified topic at the broker specified in the *Broker Address* setting.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Publish MQTT Message
Client ID: "MT Message Sender"
Broker Address: "test.mosquitto.org"
Broker Port: "8883"
Topic: "MobileTogether"
Message: "MobileTogether says Hello"
□ Use secure connection
Username:
Password:
Client Certificate: $PERSISTENT/Root/Certificate
Quality of Service: Exactly Once

The settings of the action are listed below. Most take an XPath expression that evaluates to a string.

- *Client ID:* The unique name of the MQTT publisher client.
- *Broker Address and Port:* The IP address of the broker to which the message is to be published and the port at this IP address that listens for messages. MobileTogether automatically connects to the broker when the action is triggered.
- *Topic:* The topic under which the message is being published. Subscribers to this topic at the broker will receive the message after it has been published.
- *Message:* The content of the message.
- *Secure connection:* If you want to use a secure connection, select this option and enter your username and password.
- *Client Certificate:* Enter an XPath expression that evaluates to the text of the certificate.
- *Quality of Service:* Select one of the three options allowed by [the MQTT specification]: *At most once, At least once,* or *Exactly once*.

### MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](1262) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the [Message Box](679) action. If a function is especially relevant to this action, it is listed below. For a

full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

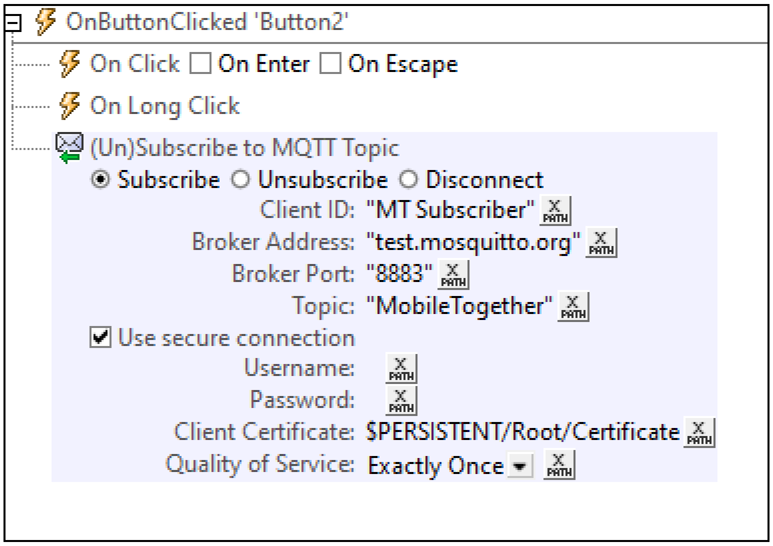
```
mt-string-to-hexBinary()
mt-hexBinary-to-string()
```

## 10.7.2 (Un)Subscribe to MQTT Topic

The (Un)Subscribe to MQTT Topic action (*screenshot below*) contains three related actions, from which you can select one via its radio button: (i) *Subscribe to a topic*; (ii) *Unsubscribe from a topic*; and (iii) *Disconnect from a broker*. These actions are described below.

### Subscribe and Unsubscribe to a topic

The *Subscribe* and *Unsubscribe* actions (*see screenshot below*) enable MT client devices to, respectively, subscribe and unsubscribe to a topic that is hosted at a particular broker. Choose the appropriate radio button (*Subscribe* or *Unsubscribe*) to set up the relevant action. The settings for both actions are the same. The *Subscribe* action, however, takes one additional setting: the *Quality of Service* setting.



OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

On Long Click

(Un)Subscribe to MQTT Topic

● Subscribe ○ Unsubscribe ○ Disconnect

Client ID: "MT Subscriber"

Broker Address: "test.mosquitto.org"

Broker Port: "8883"

Topic: "MobileTogether"

☑ Use secure connection

Username:

Password:

Client Certificate: $PERSISTENT/Root/Certificate

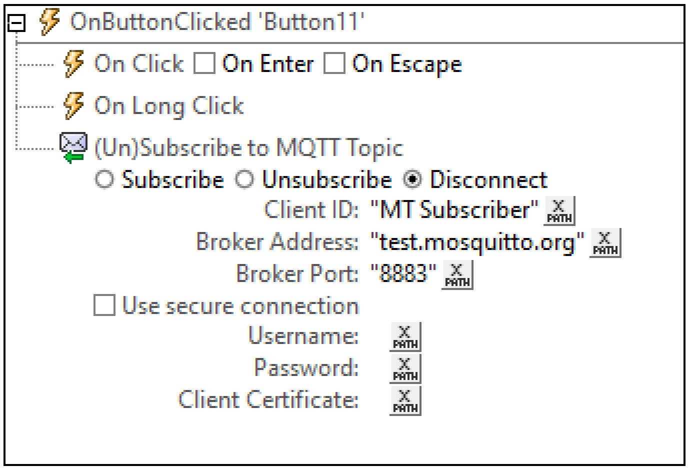
Quality of Service: Exactly Once

The settings of the *Subscribe* and *Unsubscribe* actions are listed below. They take XPath expressions that evaluate to strings.

- *Client ID:* The unique name of the MQTT subscriber client.
- *Broker Address and Port:* The IP address of the broker providing the topic for subscription and the port at this IP address via which the MT client connects. MobileTogether automatically connects to the broker when the action is triggered.
- *Topic:* The topic to which the client wants to subscribe or unsubscribe.
- *Secure connection:* If you want to use a secure connection, select this option and enter your username and password.
- *Client Certificate:* Enter an XPath expression that evaluates to the text of the certificate.
- *Quality of Service:* This setting is available for the *Subscribe* action and not for the *Unsubscribe* action. Select one of the three options allowed by the MQTT specification: *At most once, At least once,* or *Exactly once*.

## Disconnect from broker

The *Disconnect* action (*screenshot below*) disconnects the MT client device from the specified broker. Select the *Disconnect* radio button to define the action's settings.



⊟ OnButtonClicked 'Button11'

On Click □ On Enter □ On Escape

On Long Click

(Un)Subscribe to MQTT Topic

○ Subscribe ○ Unsubscribe ● Disconnect

Client ID: "MT Subscriber"

Broker Address: "test.mosquitto.org"

Broker Port: "8883"

□ Use secure connection

Username:

Password:

Client Certificate:

The settings of the *Disconnect* action are listed below. Each takes an XPath expression that evaluates to a string.

- *Client ID:* The unique name of the MQTT subscriber client.
- *Broker Address and Port:* The IP address of the broker from which the client is to be disconnected and the port at this IP address via which the client connects.
- *Secure connection:* If you want to use a secure connection, select this option and enter your username and password.
- *Client Certificate:* Enter an XPath expression that evaluates to the text of the certificate.

## MobileTogether extension functions

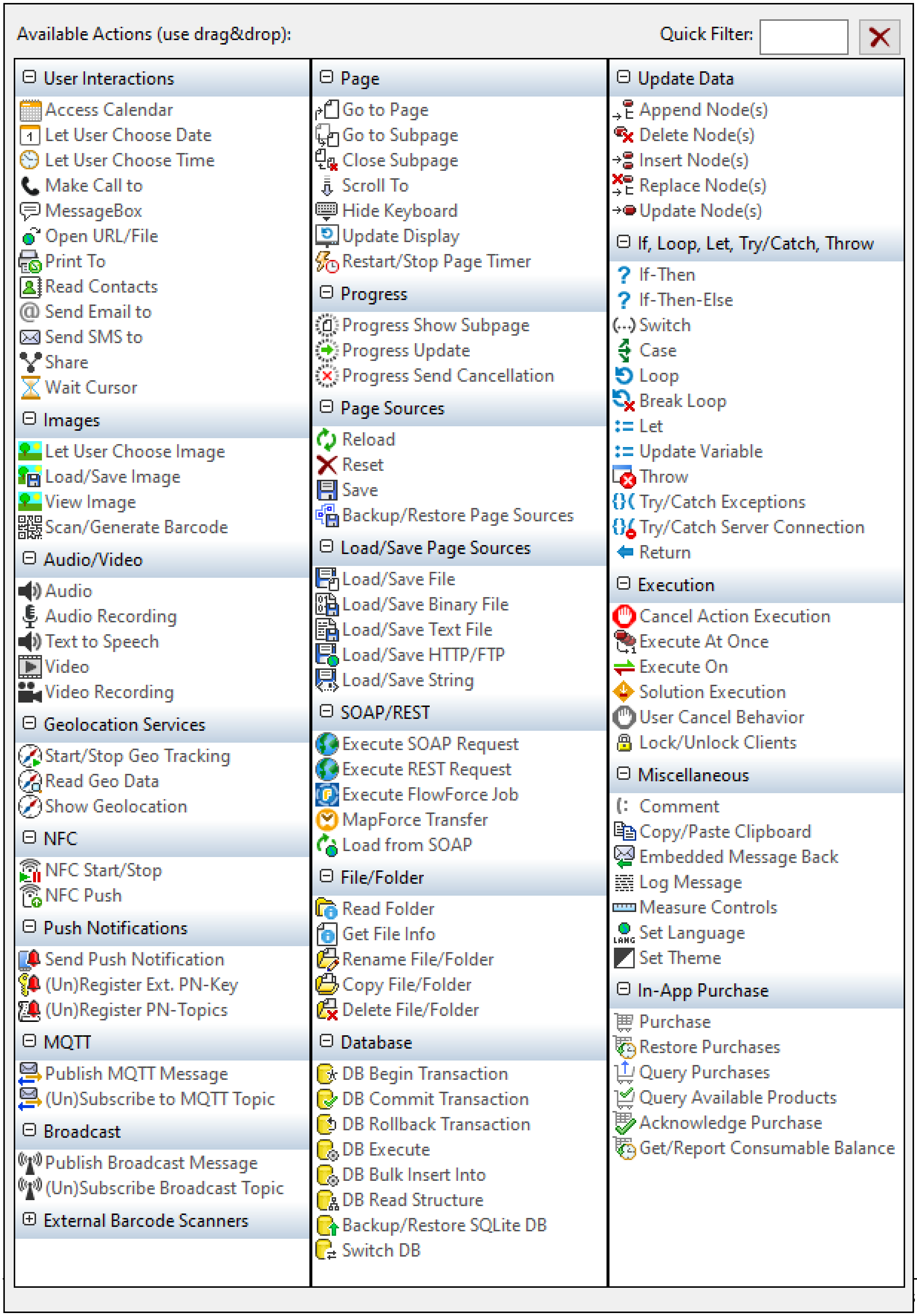
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-string-to-hexBinary()`**
**`mt-hexBinary-to-string()`**

# 10.8 Broadcast

The following actions are available in the MQTT group of the Actions dialog (*screenshot below*):

- Publish Broadcast Message 768
- (Un)Subscribe Broadcast Topic



Available Actions (use drag&drop):

Quick Filter: ✕

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

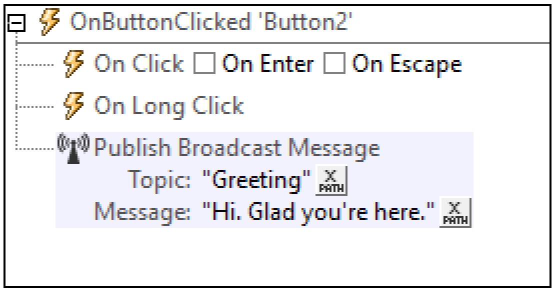
⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *[Page Events](389)* *and* *[Control Events](665)*.

See the section [Broadcasts](1142) for an overview of how you can use broadcast messages in MobileTogether solutions.

## 10.8.1 Publish Broadcast Message

The Publish Broadcast Message action (*screenshot below*) publishes the Broadcast message specified in the *Message* field to the Broadcast topic specified in the *Topic* field. Both fields take string values.



OnButtonClicked 'Button2'
On Click □ On Enter □ On Escape
On Long Click
Publish Broadcast Message
Topic: "Greeting"
Message: "Hi. Glad you're here."

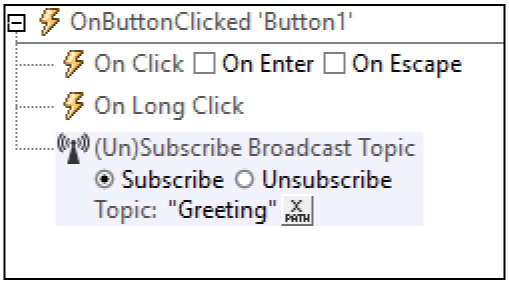
When the action is triggered, the message will be published on MobileTogether Server and will be delivered to all currently active solutions on that server that have subscribed to the specified topic.

### MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](1262) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the [Message Box](679) action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](1262).

## 10.8.2 (Un)Subscribe Broadcast Topic

The (Un)Subscribe Broadcast Topic action (*screenshot below*) enables a solution to subscribe to and unsubscribe from a Broadcast topic. Select the appropriate radio button and specify the Broadcast topic to which you want to subscribe or unsubscribe. The topic name must be a string.


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
(Un)Subscribe Broadcast Topic
● Subscribe ○ Unsubscribe
Topic: "Greeting"

Note the following points:

- From the time that the *Subscribe* action is triggered till the time that it is unsubscribed or closed or loses contact with the server, the solution will be subscribed and receive messages that are published to that topic[768].
- If the subscription ends, then it must be explicitly started with another *Subscribe* action.
- The *Unsubscribe* action causes the solution to not receive any messages published to the unsubscribed topic from the time that the Unsubscribe action is triggered. For the solution to receive messages on this topic again, a Subscribe action must be triggered.

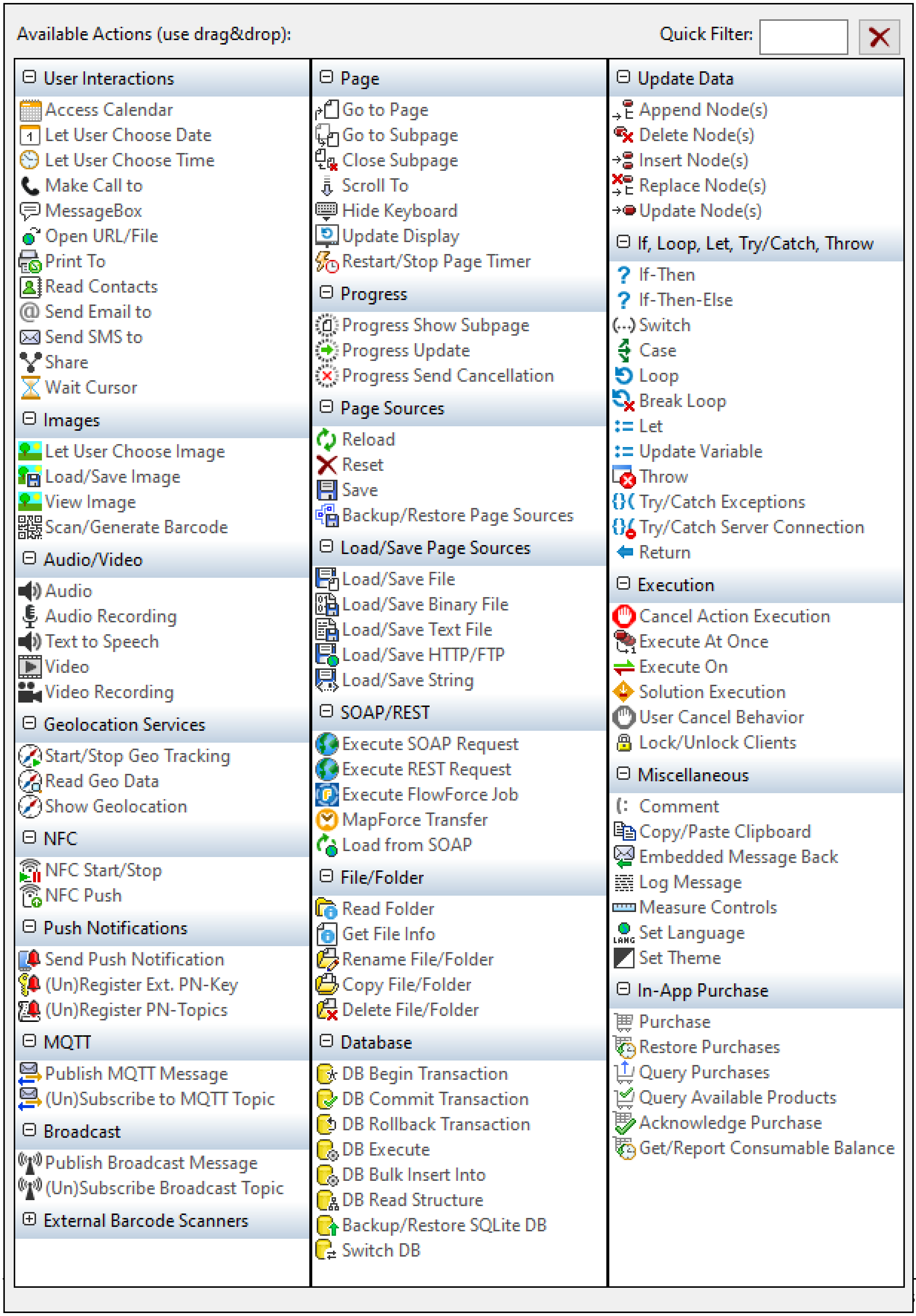
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

# 10.9 External Barcode Scanners

The External Barcode Scanners group of actions in the Actions dialog (*screenshot below*) enable you to set up the use of a barcode scanner with a MobileTogether solution.

- Zebra Connect/Disconnect 772
- Zebra Configure 773
- Zebra Mobile Computer Connect/Disconnect 775
- Zebra Mobile Computer Configure 777
- Datalogic Connect/Disconnect 778
- Datalogic Configure



Available Actions (use drag&drop):

Quick Filter:

⊟ User Interactions

- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images

- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video

- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services

- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC

- NFC Start/Stop
- NFC Push

⊟ Push Notifications

- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT

- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast

- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page

- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress

- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources

- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources

- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST

- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder

- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database

- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data

- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw

- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution

- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous

- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase

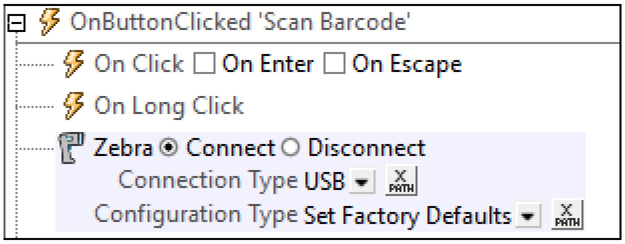
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

These actions can be triggered by page events or control events. The fastest way to access the dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events*[389] *and Control Events*[665].

Also see the section Barcode Scanners[1145] for an overview of how to use barcode scanners with your MobileTogether solution.

## 10.9.1 Zebra Connect/Disconnect

The *Zebra Connect* action and *Zebra Disconnect* action (*screenshot below*), respectively, starts and ends a connection between the MobileTogether client and a Zebra barcode scanner. Select the *Connect* or *Disconnect* radio button according to what you want to do.



⊟ OnButtonClicked 'Scan Barcode'
On Click □ On Enter □ On Escape
On Long Click
Zebra ● Connect ○ Disconnect
Connection Type USB
Configuration Type Set Factory Defaults

*Zebra Connect*

The *Zebra Connect* action connects to a Zebra scanner based on the settings you select (*listed below*). For example, if the scanner is connected via a USB cable, then select *USB* as the connection type. You can select a setting's value either from the corresponding dropdown list or by entering an XPath expression that selects a valid value. (The advantage of using an XPath expression is that a value can be selected dynamically (for example, from a node of a page source).) An XPath expression must evaluate to the string equivalent of one of the values in the corresponding dropdown list.

The action's settings are:

- *Connection Type:* Can be (i) *Bluetooth Classic/Cradle Host* (for older Android versions); (ii) *Bluetooth Low Energy (Bluetooth LE)* (for most Android and iOS devices, especially newer versions); (iii) *USB* (for Android and Windows).
- *Configuration Type.* What scanner configuration to use: (i) the scanner's current configuration; (ii) whether to set factory defaults; (iii) restore the scanner's default configuration (not different from the previous option). Note that, if the second or third option is selected (to go to the default configuration), then the scanner's configuration is reset but the connection will not be made; the connection must be made in a second and subsequent *Zebra Connect* action where *Configuration Type* is set to *Keep Current*.

When the Connect action to a Zebra scanner via Bluetooth is triggered at runtime, a dialog appears in the solution. It contains a list of available Bluetooth devices and a barcode. The end-user must select from the device list the Zebra scanner to use and then, with this scanner, scan the barcode in the solution's dialog. This starts the pairing of the MobileTogether client device with the Zebra scanner. You can use the `mt-zebra-scanner-connected()`[1262] function to check whether the Zebra scanner is connected.

Once the pairing is complete, barcodes that are scanned by the Zebra scanner will be read into the `$MT_ZEBRASCANNER` page source tree.

*Zebra Disconnect*

The *Zebra Disconnect* action ends the connection. It is advisable to end the connection after the barcode you want has been scanned. Otherwise, a scan can occur inadvertently and replace the originally scanned data in the page source tree.

### MobileTogether variables

MobileTogether provides a large number of *global variables* and *local variables*.

- **Global variables** are static: that is, their values do not change across contexts or during solution execution. For example, the value of **`$MT_CameraAvailable`**—which indicates the availability of a device camera—if `true()` at the start of solution execution will remain `true()` at all times during solution execution.
- **Local variables**, on the other hand, are dynamic. Their values can change when the execution context changes or when a device property changes. The value of the **`$MT_Portrait`** local variable, for example, can switch betwen `true()` and `false()` according to the orientation of the device.

If a MobileTogether variable is especially relevant to this action, it is listed below. For a full list of variables and their descriptions, see the topic Global Variables[1298].

**`$MT_BluetothAvailable`**
**`$MT_BluetothLEAvailable`**

### MobileTogether extension functions

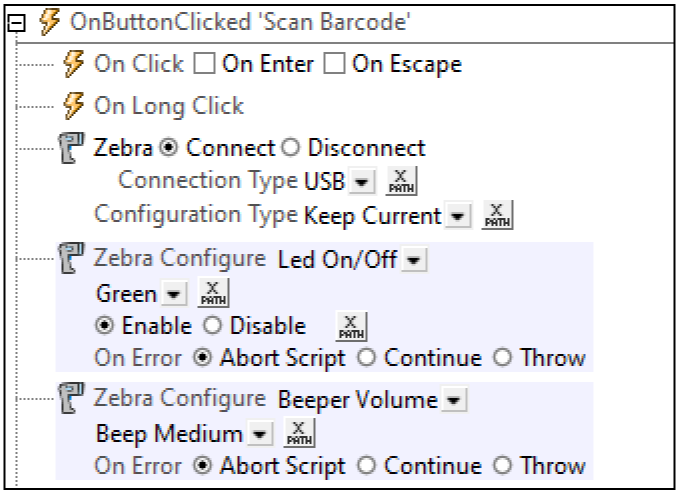
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-bluetooth-started()`**
**`mt-zebra-scanner-id()`**
**`mt-zebra-scanner-connected()`**

## 10.9.2 Zebra Configure

The *Zebra Configure* action (*screenshot below*) configures the Zebra scanner for barcode scans that are carried out subsequent to the action being triggered. Each *Zebra Configure* action defines one configuration setting. You can add as many *Zebra Configure* actions as you want to an event's action handler, and these actions will be executed in order from first to last.

**Note:** Since the MobileTogether client device must be connected to the Zebra scanner before any *Zebra Configure* action can be executed, make sure that a *Zebra Connect*[772] action is triggered before any *Zebra Configure* action is triggered.



OnButtonClicked 'Scan Barcode'
On Click □ On Enter □ On Escape
On Long Click
Zebra ● Connect ○ Disconnect
Connection Type USB
Configuration Type Keep Current
Zebra Configure Led On/Off
Green
● Enable ○ Disable
On Error ● Abort Script ○ Continue ○ Throw
Zebra Configure Beeper Volume
Beep Medium
On Error ● Abort Script ○ Continue ○ Throw

To define a configuration, select a configuration option from the *Zebra Configure* dropdown list (*see screenshot above*) and then assign to it a configuration value. You can either select a value from the corresponding dropdown list or enter an XPath expression to select a valid value. The advantage of using an XPath expression is that the value can be selected dynamically (for example, from a node of a page source). If not explicitly noted in the list below, an XPath expression must evaluate to the string equivalent of one of the values in the corresponding dropdown list.

The following Zebra scanner configuration options are available. For more information about specific configuration options, see your Zebra scanner's documentation.

- *Enable/Disable Scanner:* Available on Android and iOS. Can be used for enabling the scanner (XPath `true()`) or disabling it (XPath `false()`).
- *Beep (short, low frequency):* Available on Android and iOS. Sets the scanner for short beeps at low frequency.
- *Set Scanner Mode:* You can select whether to capture the barcode image (*Image*) or the barcode data (*Barcode*). Available on Android only and when the connection is of type *Bluetooth Classic/Cradle Host* or *USB*.
- *Beeper Duration (on scan):* Available on Android and iOS. Sets the duration of the on-scan beep.
- *Beeper Volume:* Available on Android and iOS. Sets the sound volume of the on-scan beep.
- *Beeper Frequency (on scan):* Available on Android and iOS. Sets the tonal frequency of the on-scan beep.
- *LED On/Off:* Available on Android and iOS. Sets whether the scanner's LED is enabled and, if enabled, what color the LED will be.
- *Start/Stop Tether Alarm:* Available on Android and iOS. The scanner's tether alarm alerts users when the scanner is out of range of its base station. You can set whether to start the alarm (XPath `true()`) or stop it (XPath `false()`).
- *Start/Stop Trigger:* Whether to start the use of the scanner's trigger (XPath `true()`) or to stop the use of the trigger (XPath `false()`). When trigger-use is specified, a barcode will be scanned only when the scanner's trigger is pressed. Otherwise, the trigger does not need to be pressed during a scan; in this case, a barcode will be scanned simply when placed in front of the scanner.
- *Get Device Info:* Available on Android and iOS. Gets scanner-related information. This data will be passed to the `$MT_ZEBRASCANNER` page source tree and can be accessed from there.
- *General Get/Set:* Available on Android and iOS. Allows you the flexibility of using your own XML or XML values to specify your own scanner settings and to get scanner data. You are responsible for setting this up with reference to the scanner's API.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

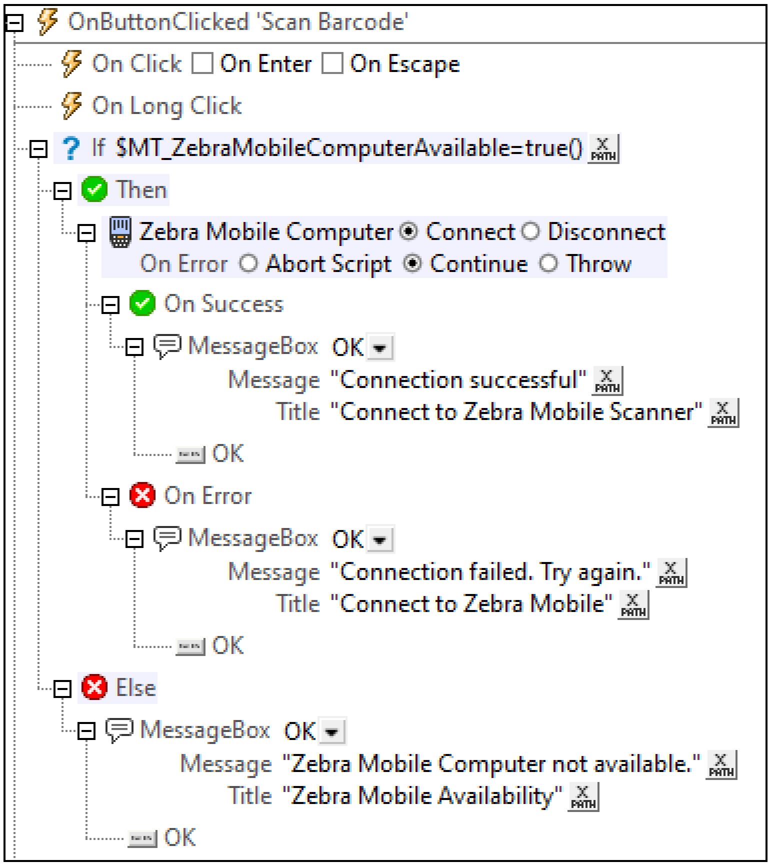
**`mt-bluetooth-started()`**
**`mt-zebra-scanner-id()`**
**`mt-zebra-scanner-connected()`**

## 10.9.3 Zebra Mobile Computer Connect/Disconnect

A Zebra mobile computer (with integrated barcode scanner) is an Android device that can also be used as a MobileTogether Client.

An MT solution running on a Zebra mobile computer can therefore easily connect to the device's barcode scanner and use the scanned barcode data. The *Zebra Mobile Computer Connect* action and *Zebra Mobile Computer Disconnect* action (*screenshot below*), respectively, starts and ends a connection between the solution and the device's barcode scanner. Select the *Connect* or *Disconnect* radio button according to what you want to do.

**Note:** You can use the **`$MT_ZebraMobileComputerAvailable`** global variable[1300] to test the availability of the Zebra mobile computer barcode scanner (*see screenshot below*).



```
OnButtonClicked 'Scan Barcode'
    On Click □ On Enter □ On Escape
    On Long Click
    If $MT_ZebraMobileComputerAvailable=true()
        Then
            Zebra Mobile Computer ● Connect ○ Disconnect
            On Error ○ Abort Script ● Continue ○ Throw
                On Success
                    MessageBox OK
                        Message "Connection successful"
                        Title "Connect to Zebra Mobile Scanner"
                        OK
                On Error
                    MessageBox OK
                        Message "Connection failed. Try again."
                        Title "Connect to Zebra Mobile"
                        OK
        Else
            MessageBox OK
                Message "Zebra Mobile Computer not available."
                Title "Zebra Mobile Availability"
                OK
```

- *Connect:* Connects the MobileTogether client app to the on-device barcode scanner. On your successfully adding the *Connect* action to the design, a `$MT_ZEBRAMOBILECOMPUTER` page source tree is added to the Page Sources Pane[270]. When a barcode is scanned with a Zebra mobile computer scanner, the barcode data will be added to this page source.
- *Disconnect:* Disconnects the MobileTogether client app from the on-device barcode scanner.

**Note:** For both actions (*Connect* and *Disconnect*), you can specify suitable actions to carry out in the event that the action succeeds (*On Success*) and in the event that it fails (*On Error*); *see screenshot above*.

**Note:** When simulating a connection, you must select *Android* | *Zebra Mobile Computer* in the Device Selector combo box in the toolbar of the designer.

## MobileTogether variables

MobileTogether provides a large number of *global variables* and *local variables*.

- **Global variables** are static: that is, their values do not change across contexts or during solution execution. For example, the value of **`$MT_CameraAvailable`**—which indicates the availability of a device camera—if `true()` at the start of solution execution will remain `true()` at all times during solution execution.
- **Local variables**, on the other hand, are dynamic. Their values can change when the execution context changes or when a device property changes. The value of the **`$MT_Portrait`** local variable, for example, can switch betwen `true()` and `false()` according to the orientation of the device.

If a MobileTogether variable is especially relevant to this action, it is listed below. For a full list of variables and their descriptions, see the topic Global Variables[1298].

**`$MT_ZebraMobileComputerAvailable`**

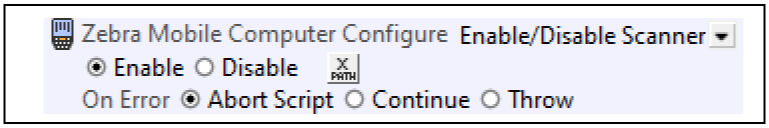
## 10.9.4 Zebra Mobile Computer Configure

The *Zebra Mobile Computer Configure* action (*screenshot below*) configures the Zebra mobile computer's (Zebra MC's) barcode scanner. New barcode scans will be carried out according to the settings defined by the *Configure* action/s. Each *Configure* action defines one configuration setting. You can add as many *Configure* actions as you want to an event's action handler, and these actions will be executed in order from first to last.

When the MobileTogether app is installed on a Zebra mobile computer, the Zebra MC assigns the MT app a default profile. If you create a new profile in the MT app (*see settings below*), then this profile will be assigned and used instead of the default profile. If you delete the MT-assigned profile, the Zebra MC will fall back on its default MT profile. For more information about profiles, see your Zebra product's technical documentation.

**Note:** Make sure that a *Zebra Mobile Computer Connect*[772] action is triggered before any *Configure* action is triggered.

**Note:** Since Zebra mobile computers are Android devices, MT solutions for them must be designed for Android.



Zebra Mobile Computer Configure Enable/Disable Scanner ▾

● Enable ○ Disable

On Error ● Abort Script ○ Continue ○ Throw

To define a configuration, select a configuration option from the *Zebra Mobile Computer Configure* dropdown list (*see screenshot above*) and then assign to it a configuration value. You can either select a value from the corresponding dropdown list or enter an XPath expression to select a valid value. The advantage of using an XPath expression is that the value can be selected dynamically (for example, from a node of a page source). If not explicitly noted in the list below, an XPath expression must evaluate to the string equivalent of one of the values in the corresponding dropdown list.

The following Zebra scanner configuration options are available. For more information about specific configuration options, see your Zebra mobile computer's technical documentation.

- *Enable/Disable Scanner:* Can be used for enabling the scanner (XPath `true()`) or disabling it (XPath `false()`).
- *Resume/Suspend Scanner:* When the scanner is enabled, you can temporarily suspend it (XPath `false()`) and resume it (XPath `true()`).
- *Start/Stop Scanning:* To start capturing barcode data, select *Start* (XPath `true()`). To stop scanning, select *Stop* (XPath `false()`).
- *Create/Delete Profile:* You can create a new profile (profile names are strings), update a profile, overwrite a profile, or delete a profile. In a profile configuration, you can also specify whether keyboard output from the Zebra mobile computer is to be enabled (XPath `true()`) or disabled (XPath `false()`). The MT-created profile will be used by the scanner instead of the scanner-created default MT profile. If you delete the MT-created profile, then the scanner falls back to its default MT profile.

- *Get Device Info:* Gets scanner-related information (*Status*, *VersionInfo*, and *ActiveProfile*). This data will be passed to the `$MT_ZEBRAMOBILECOMPUTER` page source tree and can be accessed from there.

### MobileTogether extension functions

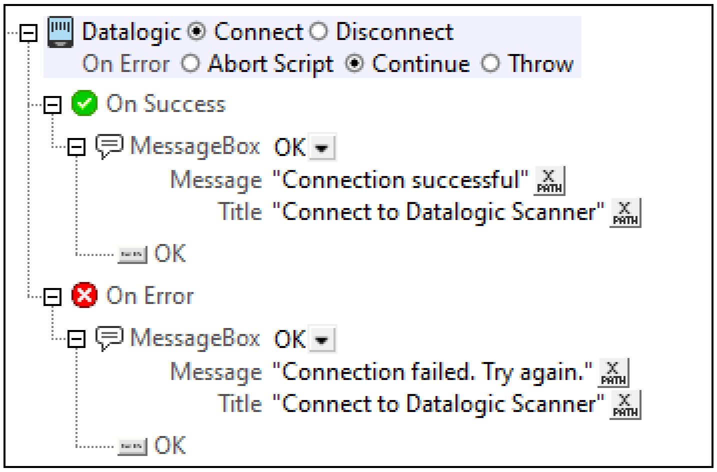
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.9.5 Datalogic Connect/Disconnect

A Datalogic barcode scanner is an Android device that can also be used as a MobileTogether Client device.

An MT solution running on a Datalogic device can therefore easily connect to the device's barcode scanner and store and use the scanned barcode data. The *Datalogic Connect* action and *Datalogic Disconnect* action (*screenshot below*), respectively, starts and ends a connection between the solution and the device's barcodescanner. Select the *Connect* or *Disconnect* radio button according to what you want to do.

**Note:** You can use the **`$MT_DatalogicScannerAvailable`** global variable[1300] to test the availability of the Datalogic barcode scanner.



```
Datalogic ● Connect ○ Disconnect
  On Error ○ Abort Script ● Continue ○ Throw
  On Success
    MessageBox OK
        Message "Connection successful"
          Title "Connect to Datalogic Scanner"
      OK
  On Error
    MessageBox OK
        Message "Connection failed. Try again."
          Title "Connect to Datalogic Scanner"
      OK
```

- *Connect:* Connects the MobileTogether client app to the on-device barcode scanner. On your adding the *Connect* action to the design, a `$MT_DATALOGICSCANNER` page source tree is added to the Page Sources Pane[270]. When a barcode is scanned with a Datalogic scanner, the barcode data will be added to this page source.
- *Disconnect:* Disconnects the MobileTogether client app from the on-device barcode scanner.

**Note:** For both actions (*Connect* and *Disconnect*), you can specify suitable actions to carry out in the event that the action succeeds (*On Success*) and in the event that it fails (*On Error*); *see screenshot above*.

**Note:** When simulating a connection, you must select *Android* | *Datalogic* in the Device Selector combo box in the toolbar of the designer.

### MobileTogether variables

MobileTogether provides a large number of *global variables* and *local variables*.

- **Global variables** are static: that is, their values do not change across contexts or during solution execution. For example, the value of **`$MT_CameraAvailable`**—which indicates the availability of a device camera—if `true()` at the start of solution execution will remain `true()` at all times during solution execution.
- **Local variables**, on the other hand, are dynamic. Their values can change when the execution context changes or when a device property changes. The value of the **`$MT_Portrait`** local variable, for example, can switch betwen `true()` and `false()` according to the orientation of the device.

If a MobileTogether variable is especially relevant to this action, it is listed below. For a full list of variables and their descriptions, see the topic Global Variables 1298.

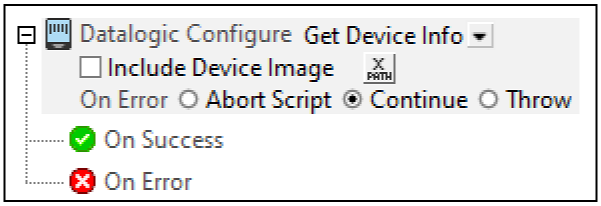
**`$MT_DatalogicScannerAvailable`**

## 10.9.6 Datalogic Configure

The *Datalogic Configure* action (*screenshot below*) configures the Datalogic barcode scanner. New barcode scans will be carried out according to the settings defined by the *Configure* action/s. Each *Configure* action defines one configuration setting. You can add as many *Configure* actions as you want to an event's action handler, and these actions will be executed in order from first to last.

**Note:** Make sure that a *Datalogic Connect/Disconnect* 778 action is triggered before any *Configure* action is triggered.

**Note:** Since Datalogic scanners are Android devices, MT solutions for them must be designed for Android.



To define a configuration, select a configuration option from the *Datalogic Configure* dropdown list (*see screenshot above*) and then assign to it a configuration value. You can either select a value from the corresponding dropdown list or enter an XPath expression to select a valid value. The advantage of using an XPath expression is that the value can be selected dynamically (for example, from a node of a page source). If not explicitly noted in the list below, an XPath expression must evaluate to the string equivalent of one of the values in the corresponding dropdown list.

The following Datalogic scanner configuration options are available. For more information about specific configuration options, see your Datalogic scanner's technical documentation.

- *Start/Stop Decode:* To start decoding barcode data, select *Start* (XPath `true()`). To stop decoding, select *Stop* (XPath `false()`).

- *Press/Release Trigger:* Whether to start the decoding on the press of the trigger (XPath `true()`) or the release of the trigger (XPath `false()`).
- *Enable/Disable Keyboard Wedge:* Whether the keyboard wedge feature of Datalogic scanners is enabled (XPath `true()`) or disabled (XPath `false()`).
- *Set LED On/Off/Blink:* Sets the LED mode and what triggers the LED.
- *Get Device Info:* Gets scanner-related information and lets you choose whether to include the device image or not. This data will be passed to the `$MT_DATALOGICSCANNER` page source tree and can be accessed from there.

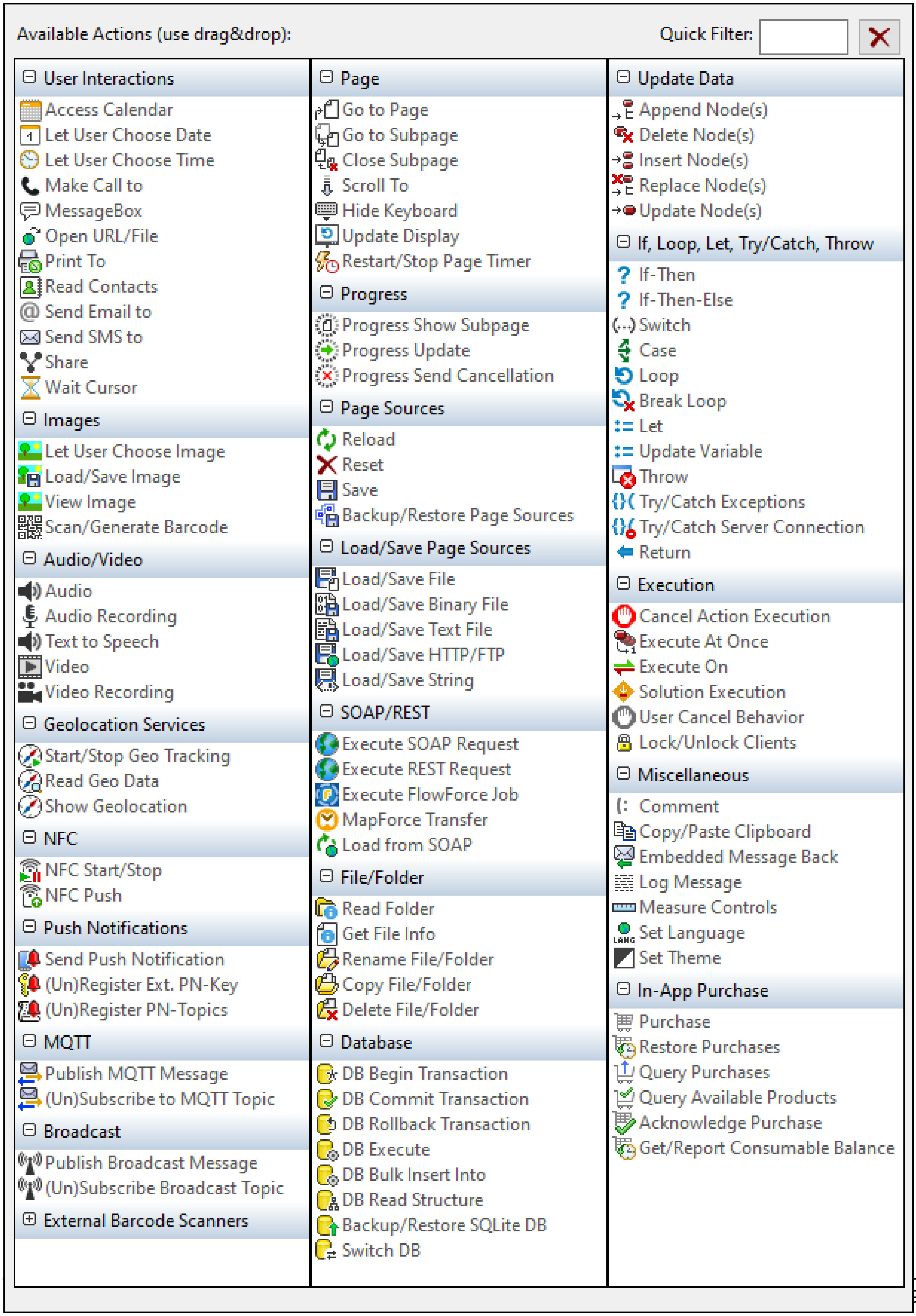
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.10 Page

The following actions are available in the Page group of the Actions dialog:

- Go to Page 783
- Go to Subpage 783
- Close Subpage 788
- Scroll To 788
- Hide Keyboard 790
- Update Display 790
- Restart/Stop Page Timer



Available Actions (use drag&drop):

Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

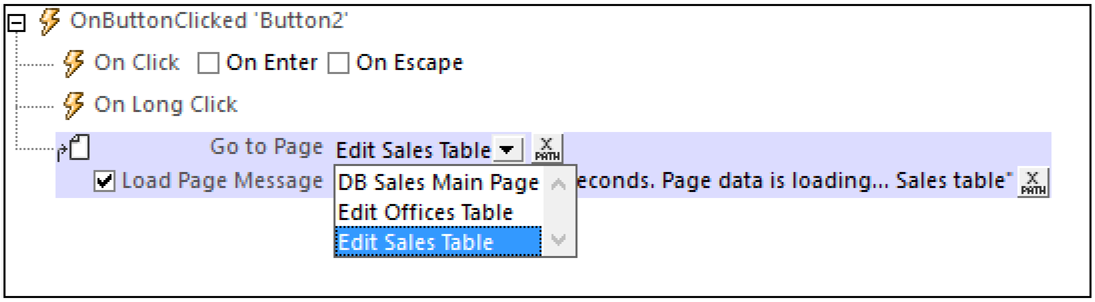
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also [Page Events](389)* [389] *and [Control Events](665)* [665].

## 10.10.1 Go to Page

When triggered, the `GoToPage` action goes to the specified page. The page to go to is selected in the combo box of the action (*see screenshot below*) or via an XPath expression. If no other page exists in the project, no page will be available for selection in the combo box. In the screenshot below, the `GoToPage` action is placed below the two sub-events `On Click` and `On Long Click`. This defines that the action is triggered when the button is either tapped (clicked briefly) or pressed (clicked for a longer time).



OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

On Long Click

Go to Page: Edit Sales Table

☑ Load Page Message: econds. Page data is loading... Sales table"

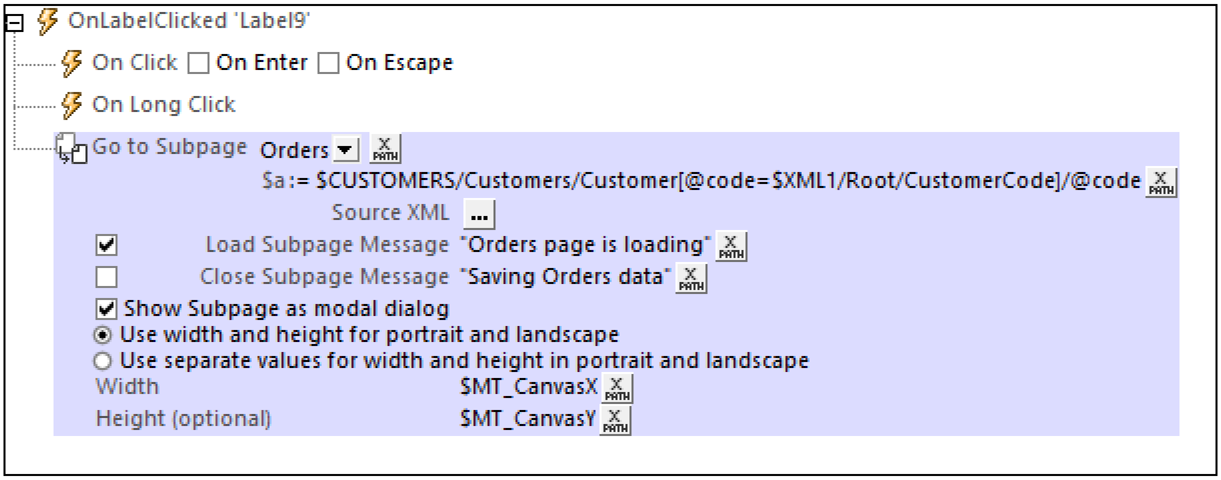
The *Load Page Message* option (*see screenshot above*) enables a message to be shown in a progress dialog while the page loads. If you want to display a message, check this option, and then define the message as an XPath expression. If the option is unchecked, no message will be displayed.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions [1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box [679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions [1262].

## 10.10.2 Go to Subpage

Goes to the specified sub-page when triggered. The subpage to go to is selected from the *Go to Subpage* combo box (*see screenshot below*), which lists all the subpages in the project. In the screenshot below, an `OnLabelClicked` event has been set to go to the *Orders* subpage of the MobileTogether project. Alternatively, the name of the subpage can be obtained via an XPath expression, for example: `"Orders" or $XML3/DataSources/Subpage[@id="Name"]/@name`.



OnLabelClicked 'Label9'
On Click □ On Enter □ On Escape
On Long Click
Go to Subpage Orders
$a:= $CUSTOMERS/Customers/Customer[@code=$XML1/Root/CustomerCode]/@code
Source XML ...
☑ Load Subpage Message "Orders page is loading"
□ Close Subpage Message "Saving Orders data"
☑ Show Subpage as modal dialog
● Use width and height for portrait and landscape
○ Use separate values for width and height in portrait and landscape
Width $MT_CanvasX
Height (optional) $MT_CanvasY

The Go to Subpage action has the following settings:

- *Parameters:* If parameters have been declared on the selected sub page[381], then each parameter of that sub page is now listed below the sub page entry, and a value can be passed to each parameter. In the screenshot above, for example, a value is defined for the parameter named **$a**. It is an error if no value is passed to a mandatory parameter. If the sub page is not selected via the combo box but by using an XPath expression, then the name of the sub page is not generated till runtime. Since no parameters can, therefore, be known at design time, instead of any parameter names being displayed, an entry for *Parameters* is displayed. Here, you can enter an XPath map expression or array expression to define parameter values, for example: `map{"a": "a-value", "second-param": "second-param-value"}` or `[("a-value"), ("second-param-value")]`. If you enter an array expression, (i) the order of sequences in the array must match the order of parameter declarations of the sub page, and (ii) values for optional parameters may not be left out of the array (an empty sequence can be entered for a parameter value that you wish to leave empty). Note that you can use other parameters when defining the value of a parameter.
- *Source XML:* Specifies whether and how subpage data should correspond to page sources in the originating page. This definition is entered in the Subpage Data Mapping dialog, which is accessed by clicking the **Additional Dialog** button of the *Source XML* field. *See the section "Source XML of subpages*[785]*" below for a detailed description of this dialog.*
- *Load Subpage Message:* In the event that the subpage is transmitted via a server request: If the check box is selected, this message is shown in a progress dialog while the subpage loads. *See screenshot above*.
- *Close Subpage Message:* In the event that the subpage is transmitted via a server request: If the check box is selected, this message is shown in a progress dialog while the subpage closes. *See screenshot above*.
- *Show Subpage as modal dialog:* If this option is **unchecked**, then the top page in the display is overlaid by the subpage; (the user can return to the top page by tapping the **Back** button). If this option is **checked**, then the subpage is opened in a separate window above the top page (known as a modal dialog)—with access to top-page content being disabled. If the user taps outside the modal window, then the actions specified for the `OnBackButtonClicked`[393] page-event are executed. If no action is specified for this event, then the subpage is closed and execution returns to the top page.

*Defining modal dialogs*
The following definitions can be applied to modal dialogs:

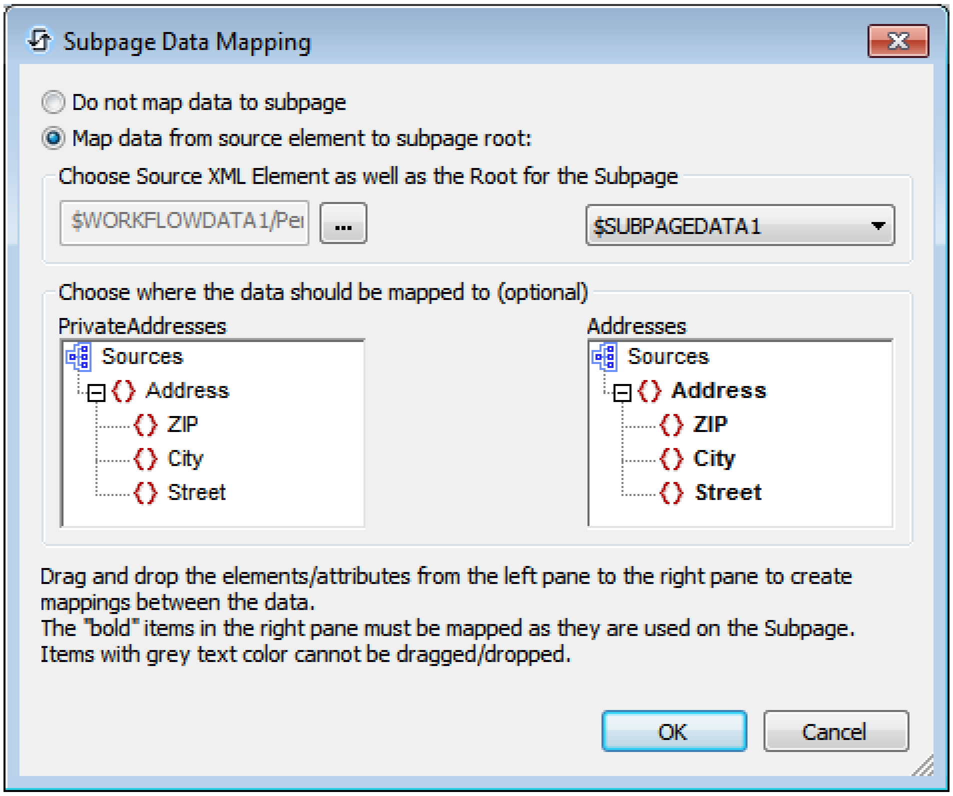
- You can set the dimensions of the modal dialog window via the XPath expressions of the *Width* and *Height* settings.
- If the optional *Height* setting is not specified, then the height will be only as much as is required to fit the content, up to a maximum of the device's viewport height. If a greater height is required, then a scrollbar is automatically added to the modal dialog.
- Dimension values that you enter may be a number (for pixel dimensions) or a string that contains a number value and a % sign (to specify the dialog's dimension as a percentage of the viewport's dimension).
- In the screenshot above, the dimensions have been set to the device-specific width and height values (viewport dimensions) 1304 that are obtained dynamically from the device at run time.
- You can choose whether the modal dialog should: (i) have the same single set of dimensions for both portrait and landscape orientations, or (ii) use a different set of dimensions for each orientation (that is, one set of dimensions for portrait and another for landscape).
- If the modal dialog is defined to have a single set of dimensions for both orientations, then the width and height dimensions are taken from the portrait orientation of the page that was current before the *Go to Subpage* action was executed. If this option was selected, then the modal dialog will have the same set of width and height values in both orientations. The effect is that the modal dialog does not change size when the orientation is changed.
- From a subpage that is displayed as a modal dialog, you can go to another subpage, which you can also choose to open as a modal dialog above the previous modal dialog.
- Relative dimensions (percentage values) of all modal dialogs—even when they originate from other modal dialogs—are calculated relative to the dimensions of the **device's viewport**.
- If a subpage is opened from a modal dialog without the modal dialog option being selected, then the new subpage will have the same dimensions as the modal dialog and will overlay it. It will look as if the new subpage has replaced the previous subpage.

## Source XML of subpages

Every subpage must have at least one page source in order for the subpage to be used effectively. Data in page sources of subpages can be mapped to data in top page sources, or the data in subpage sources can be left unmapped. If mapped, then subpage data is transferred to the mapped nodes of the top page when the subpage is closed. The behavior of the source XML of subpages is defined in the Subpage Data Mapping dialog (*screenshot below left*). To access this dialog, click the **Additional Dialog** button of the *Source XML* field of the Go to Subpage action definition (*see description above*).

The following subpage data handling possibilities are available:

- Subpage data is not mapped. In this case, page sources of subpages are handled independently of the subpage-data-mapping mechanism.
- A subpage source is mapped to a top page structure that is exactly the same; even the names of elements and attributes are the same.
- A subpage source is mapped to a top page source that has a different structure and/or different attribute/element names.



Subpage Data Mapping

○ Do not map data to subpage

● Map data from source element to subpage root:

Choose Source XML Element as well as the Root for the Subpage

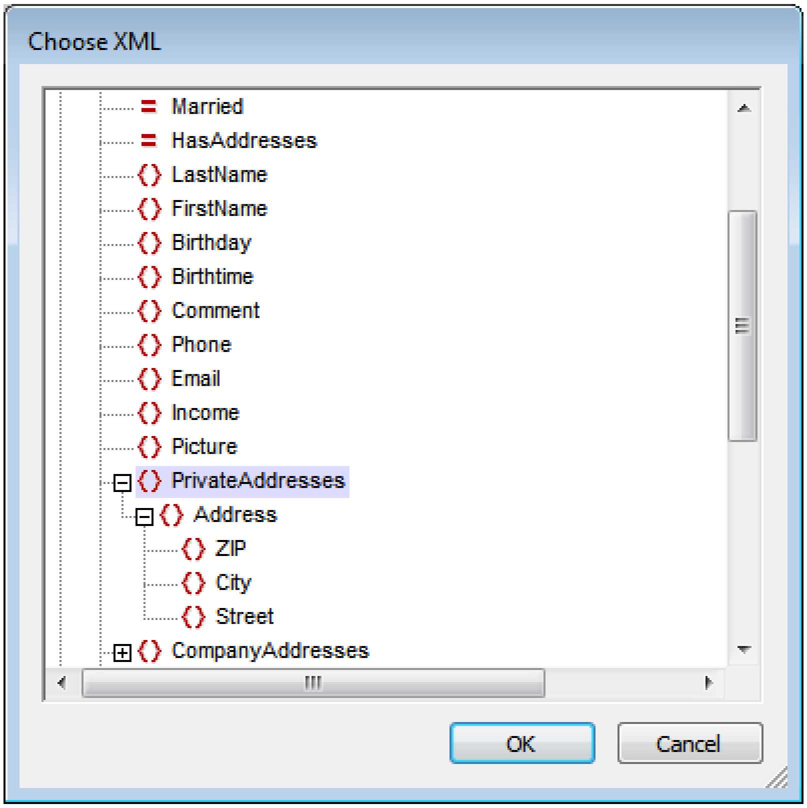
$WORKFLOWDATA1/Pe

$SUBPAGEDATA1

Choose where the data should be mapped to (optional)

| PrivateAddresses | Addresses |
|---|---|
| Sources | Sources |
| Address | **Address** |
| ZIP | **ZIP** |
| City | **City** |
| Street | **Street** |

Drag and drop the elements/attributes from the left pane to the right pane to create mappings between the data.
The "bold" items in the right pane must be mapped as they are used on the Subpage.
Items with grey text color cannot be dragged/dropped.



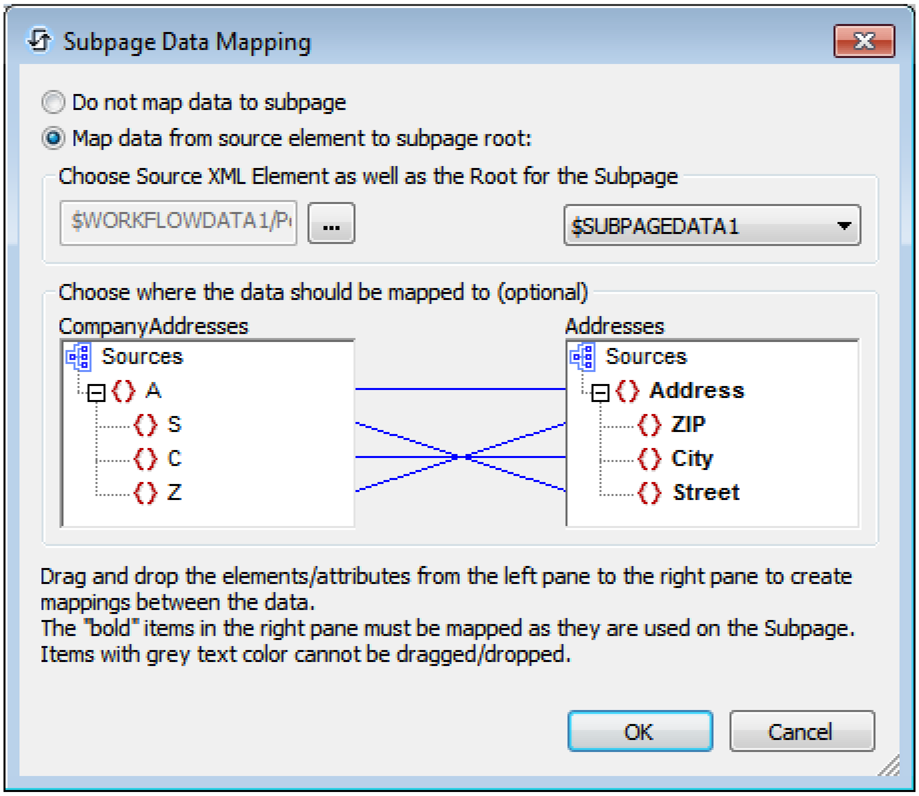
Choose XML

```
= Married
= HasAddresses
{} LastName
{} FirstName
{} Birthday
{} Birthtime
{} Comment
{} Phone
{} Email
{} Income
{} Picture
⊟ {} PrivateAddresses
   ⊟ {} Address
        {} ZIP
        {} City
        {} Street
⊞ {} CompanyAddresses
```

In the Subpage Data Mapping dialog (*screenshot above left*), the page source tree of the subpage is displayed in the right-hand source tree pane, while the page source tree of the top page is displayed in the left-hand source tree pane .

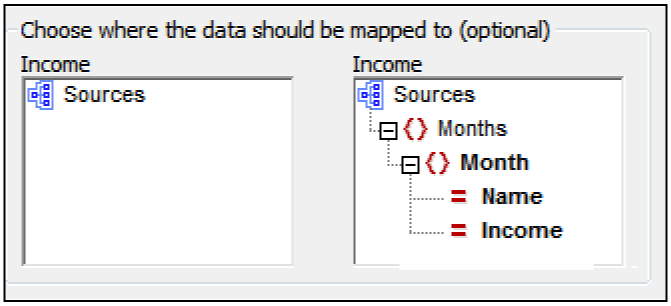
- Select the subpage page source from the combo box above the (right-hand) subpage pane. This combo box lists all the page sources of the subpage. Note that the subpage name is listed above the subpage pane.
- To select the top-page page source to be mapped, first click the **Additional Dialog** button above the (left-hand) top page pane. This displays the Choose XML dialog (*screenshot above right*), which contains all the page sources of the top page. Select the node that you want to map to the subpage data. In the screenshot above, the `PrivateAddresses` node has been selected in the Choose XML dialog. As a result, the descendants of `PrivateAddresses` are displayed in the top page source tree pane and *PrivateAddresses*, the mapped node, is listed above the pane.

In the case described by the screenshot above, the mapping is automatically done between the two page sources since both XML data structures are identical, right down to the names of elements. In the screenshot below a mapping between elements is constructed by dragging a node from the left-hand pane onto a node in the right-hand pane. This is because the two data structures are not identical (the first element, `S`, must correspond to the third element, `Street`).


Subpage Data Mapping
○ Do not map data to subpage
● Map data from source element to subpage root:
Choose Source XML Element as well as the Root for the Subpage
$WORKFLOWDATA1/P
...
$SUBPAGEDATA1
Choose where the data should be mapped to (optional)
CompanyAddresses
Sources
A
S
C
Z
Addresses
Sources
Address
ZIP
City
Street
OK
Cancel

Drag and drop the elements/attributes from the left pane to the right pane to create mappings between the data.
The "bold" items in the right pane must be mapped as they are used on the Subpage.
Items with grey text color cannot be dragged/dropped.

In the case described by the screenshot below, the content of the subpage data node `Income/Months/Month` is mapped to the top page data node `Income` (indicated by the title of the pane).


Choose where the data should be mapped to (optional)

Income
Sources

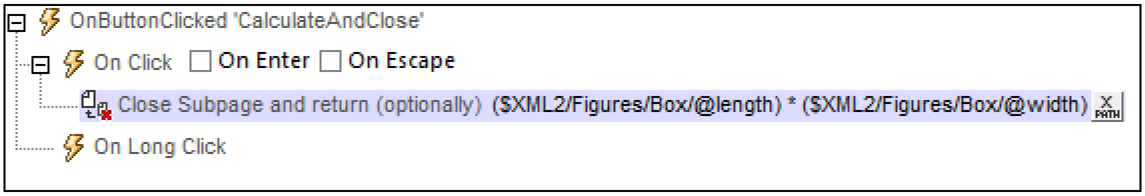
Income
Sources
Months
Month
Name
Income

## MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](1262) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the [Message Box](679) action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](1262).

# 10.10.3 Close Subpage

Closes the active subpage when triggered. Optionally, when the subpage is closed, a return value can be saved. The return value is generated by an XPath expression that you specify (*see screenshot below*). The value can be retrieved as the Subpage Result of the [Let action](900).


OnButtonClicked 'CalculateAndClose'
On Click □ On Enter □ On Escape
Close Subpage and return (optionally) ($XML2/Figures/Box/@length) * ($XML2/Figures/Box/@width)
On Long Click

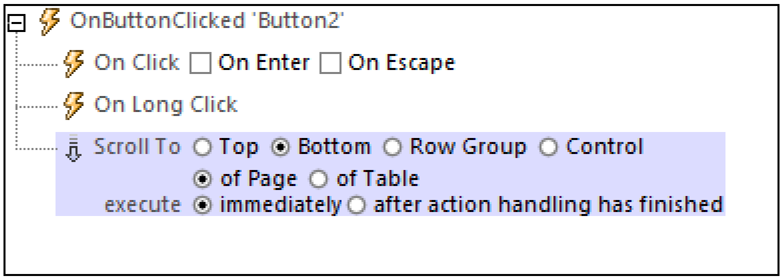
## MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](1262) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the [Message Box](679) action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](1262).

# 10.10.4 Scroll To

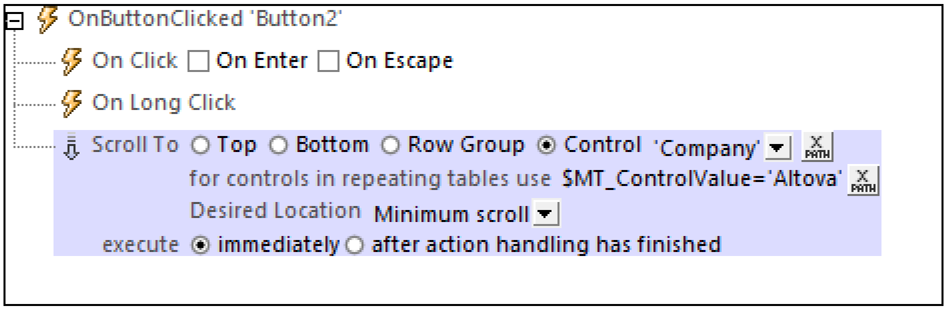
You can specify that, when the action is triggered, the solution scrolls to one of the following:

- Top or bottom of the active page (*screenshot below*).



OnButtonClicked 'Button2'
On Click □ On Enter □ On Escape
On Long Click
Scroll To ○ Top ● Bottom ○ Row Group ○ Control
● of Page ○ of Table
execute ● immediately ○ after action handling has finished

- Top or bottom of the selected table. The table is selected by choosing its name from the combo box or by entering an XPath expression that evaluates to the table's name. Only tables that have vertical scrolling[1085] are listed for selection in the *Table* option's combo box.
- A row-group of a table that is identified by name. The table's name is selected from a combo box or entered as an XPath expression. You can optionally specify an XPath condition to select a specific row group (of the selected table). The first row group for which the condition evaluates to **`true`** will be the target of the Scroll To action.
- A control that you select by its name (*see screenshot below*). Select the name from the combo box or create an XPath expression that evaluates to the name of the control. If you select a control in a repeating-table structure, then you can specify an additional condition, via an XPath expression, to refine the search for the control. The dynamic variables that return control information[1304], such as **`$MT_ControlValue`**, can be used to locate a control in a dynamically changing context.



OnButtonClicked 'Button2'
On Click □ On Enter □ On Escape
On Long Click
Scroll To ○ Top ○ Bottom ○ Row Group ● Control 'Company'
for controls in repeating tables use $MT_ControlValue='Altova'
Desired Location Minimum scroll
execute ● immediately ○ after action handling has finished

*Desired location in the view*

The *Desired Location* combo box becomes available when the *Row Group* or *Control* option is selected. It enables you to specify where in the view the target object must be shown. A value of *Minimum Scroll* brings the target into view with the minimum of scrolling. So, for example, if you scroll to an Image control, minimum scroll will just bring the full image into view. A value of *Top* scrolls the target so that it is at the top of the view. *Center* scrolls the target to the center of the view, and *Bottom* scrolls it to the bottom of the view.

*Execute (immediately or after action-handling)*

The *Execute* option allows you to define whether the Scroll-To action is executed (i) immediately upon being processed, or (ii) after the processing of all actions of the current event have completed. The delayed processing can be useful, for example, if subsequent actions generate the page component that is the target of the Scroll-To action.

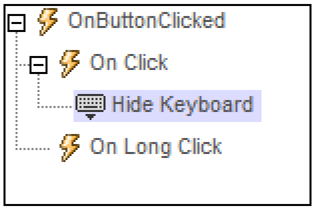
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a

full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.10.5 Hide Keyboard

Hides the keyboard of the mobile device when the action is triggered. Some mobile devices automatically show a keyboard when certain page elements, such as text fields, are present or active on the page. The Hide Keyboard action is useful if you wish to gain some screen space for the page display on such devices.

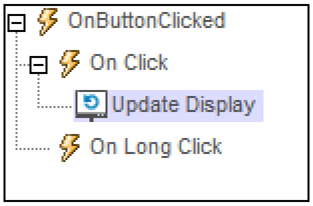


### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.10.6 Update Display

Updates all the controls of a page with the data currently contained in each control's corresponding page source node. This enables the display explicitly to be updated immediately after an action is executed. For example, if the Update Display action is used inside a loop action 897, then the display is updated when the actions in each loop iteration are executed.



*Updating selected controls/nodes*

By default, all the controls of a page are updated with data held in the corresponding page source nodes. You can, however, update only a selection of controls. The selection is done by specifying, via XPath, the page source nodes that provide the data of these controls.

To update only a selection of nodes, do the following:

1. Go to the More Project Settings dialog[296] (by double-clicking the value field of the `More Project Settings` property in the *Project* section of the Styles & Properties Pane[274]).
2. Set *Advanced UpdateDisplay Options* to `true` and close the dialog.
3. If you now add an Update Display action to any event, the action will provide the four options listed below. Select the option you want
   - *All* updates all controls on the page (with data currently contained in the corresponding page source nodes). This is equivalent to the default setting when advanced options are disabled *(see above)*.
   - *All Except* updates all controls on the page except those that are linked to the nodes selected by the option's XPath expression.
   - *Only* updates only those controls that are linked to the nodes selected by the option's XPath expression.
   - *None* updates no control and the display remains unchanged.

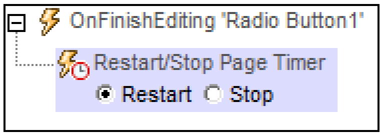
**Note:** When the Update Display action is executed in a control or page context, only the variables currently available to the control or page, respectively, can be executed. Variables inside an Action Group[940] will not be available outside the Action Group, and so cannot be used in the XPath expressions to select controls/nodes to update *(see list above)*. If you want to use a variable in these XPath expressions, define the variable at a global level or set its value as the content of a node in a page source[315] (for example, in the **`$PERSISTENT`** page source).

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.10.7 Restart/Stop Page Timer

The page timer is configured in the **`OnPageRefresh`**[390] event tab, and it is used to time the intervals between page refreshes. The timer's initial refresh interval is specified in the timer configuration. When the page is loaded, the timer is started, and the page is refreshed at the specified refresh interval.



- *Restart Page Timer:* If the timer's refresh interval is changed at some subsequent point during page processing, then the timer must be restarted. This is required to re-initialize the timer with the new refresh interval. *See the SOAP Requests tutorial*[226] *for an example*.
- *Stop Page Timer:* When a page is refreshed, the actions defined in the On Timer Refresh action[390] are executed. If the timer continues to run, the page will be refreshed regularly at the specified interval, and the refresh actions will be executed each time. If you wish to stop these actions being executed at some later point during page processing, use the Stop Page Timer action. This stops the timer—and, consequently, stops the actions.

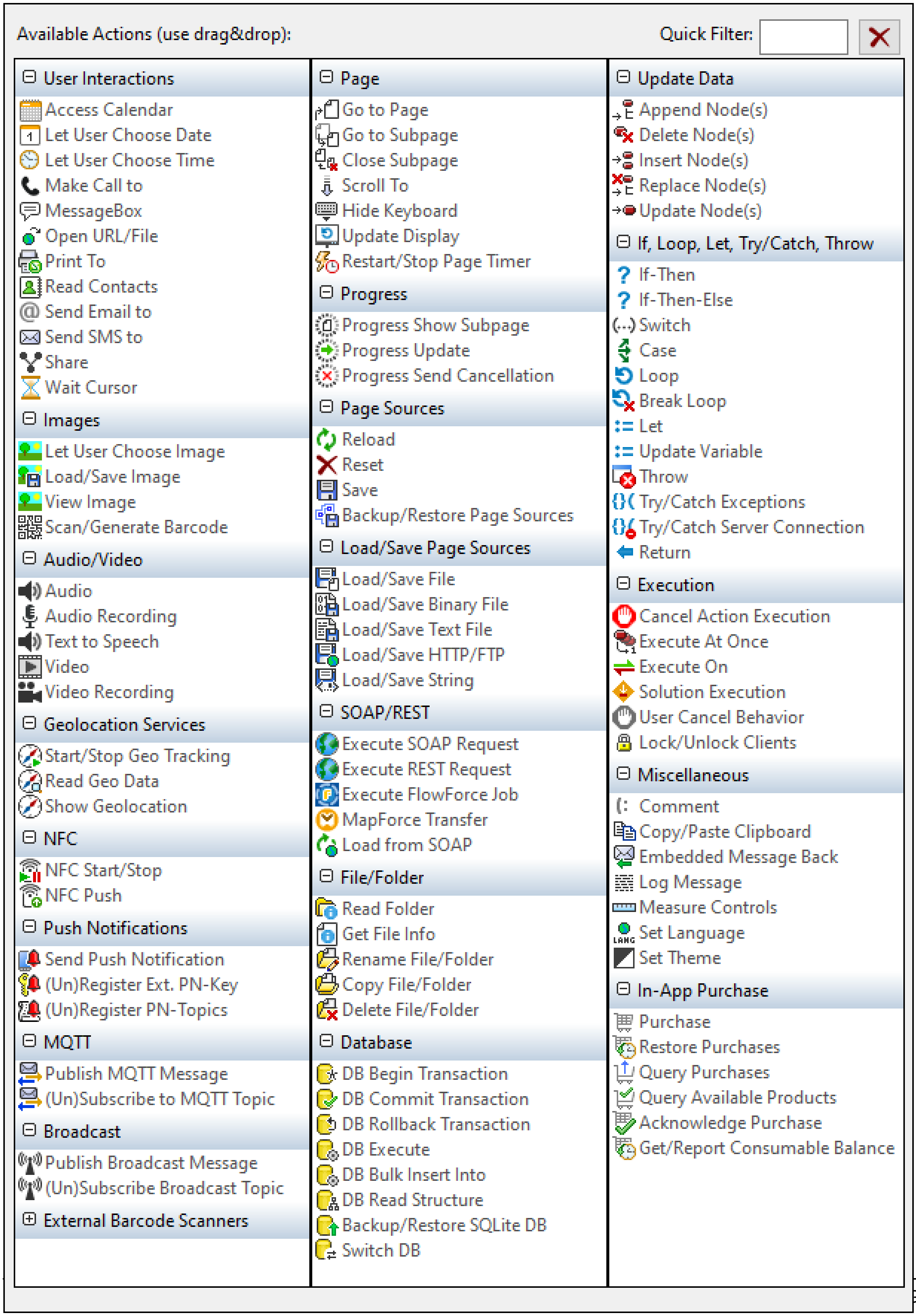
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

# 10.11 Progress

The following actions are available in the Progress group of the Actions dialog:

- Progress Show Subpage[795]
- Progress Update[796]
- Progress Send Cancellation



Available Actions (use drag&drop):

Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

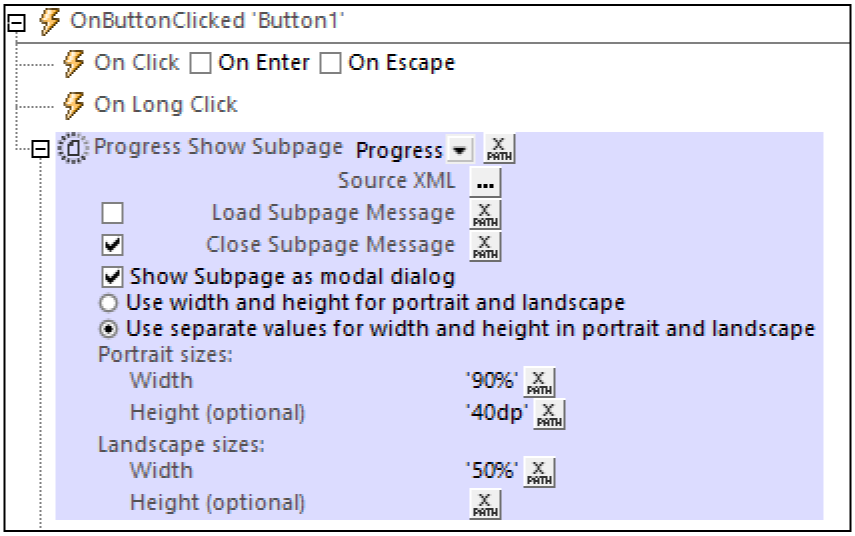
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events* 389 *and* *Control Events* 665.

## 10.11.1 Progress Show Subpage

The Progress Show Subpage action (*screenshot below*) is used (i) to specify the subpage that will display the progress of server actions; and (ii) to define, as its child actions, the server actions to carry out (for which a progress indicator is required).



⊟ OnButtonClicked 'Button1'

On Click ☐ On Enter ☐ On Escape

On Long Click

⊟ Progress Show Subpage Progress

Source XML ...

☐ Load Subpage Message

☑ Close Subpage Message

☑ Show Subpage as modal dialog

○ Use width and height for portrait and landscape

● Use separate values for width and height in portrait and landscape

Portrait sizes:

Width '90%'

Height (optional) '40dp'

Landscape sizes:

Width '50%'

Height (optional)

### Subpage properties

- The dropdown list of the action lists the subpages of the project. Select the subpage you want to use to display the progress of the server actions. In the screenshot above, the *Progress* subpage has been selected. Alternatively, you can enter an XPath expression that will resolve to the name of the target subpage. This is useful if you want to select the subpage dynamically (that is, if selection is dependent on the runtime situation).
- *Source XML:* Specifies whether and how subpage data should correspond to page sources in the originating page. This definition is entered in the Subpage Data Mapping dialog, which is accessed by clicking the **Additional Dialog** button of the *Source XML* field. *See the section "Source XML of subpages* 785*" below for a detailed description of this dialog.*
- *Load Subpage Message:* In the event that the subpage is transmitted via a server request: If the check box is selected, this message is shown in a progress dialog while the subpage loads. *See screenshot above*.
- *Close Subpage Message:* In the event that the subpage is transmitted via a server request: If the check box is selected, this message is shown in a progress dialog while the subpage closes. *See screenshot above*.
- *Show Subpage as modal dialog:* If this option is **unchecked**, then the top page in the display is overlaid by the subpage; (the user can return to the top page by tapping the **Back** button). If this option is **checked**, then the subpage is opened in a separate window above the top page (known as a modal

dialog)—with access to top-page content being disabled. If the user taps outside the modal window, then the actions specified for the `OnBackButtonClicked` 393 page-event are executed. If no action is specified for this event, then the subpage is closed and execution returns to the top page.

*Defining modal dialogs*
The following definitions can be applied to modal dialogs:

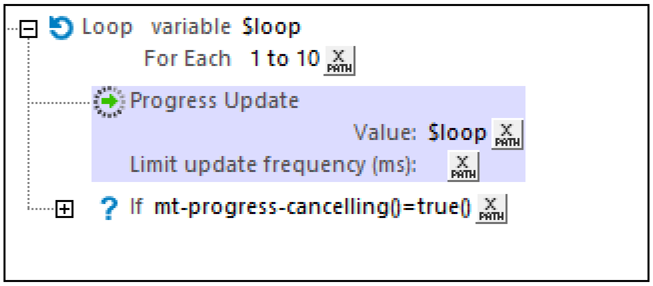
- You can set the dimensions of the modal dialog window via the XPath expressions of the *Width* and *Height* settings.
- If the optional *Height* setting is not specified, then the height will be only as much as is required to fit the content, up to a maximum of the device's viewport height. If a greater height is required, then a scrollbar is automatically added to the modal dialog.
- Dimension values that you enter may be a number (for pixel dimensions) or a string that contains a number value and a % sign (to specify the dialog's dimension as a percentage of the viewport's dimension).
- In the screenshot above, the dimensions have been set to the device-specific width and height values (viewport dimensions) 1304 that are obtained dynamically from the device at run time.
- You can choose whether the modal dialog should: (i) have the same single set of dimensions for both portrait and landscape orientations, or (ii) use a different set of dimensions for each orientation (that is, one set of dimensions for portrait and another for landscape).
- If the modal dialog is defined to have a single set of dimensions for both orientations, then the width and height dimensions are taken from the portrait orientation of the page that was current before the *Go to Subpage* action was executed. If this option was selected, then the modal dialog will have the same set of width and height values in both orientations. The effect is that the modal dialog does not change size when the orientation is changed.
- From a subpage that is displayed as a modal dialog, you can go to another subpage, which you can also choose to open as a modal dialog above the previous modal dialog.
- Relative dimensions (percentage values) of all modal dialogs—even when they originate from other modal dialogs—are calculated relative to the dimensions of the **device's viewport**.
- If a subpage is opened from a modal dialog without the modal dialog option being selected, then the new subpage will have the same dimensions as the modal dialog and will overlay it. It will look as if the new subpage has replaced the previous subpage.

## Server actions

A progress indicator provides progress information about a particular set of actions. This set is comprised of the child actions of the Progress Show Subpage action. For an example of how this works, see the usage of the Progress Show Subpage action 242 in the Progress Indicator tutorial 241.

## 10.11.2 Progress Update

The Progress Update action (*see screenshot below*) is used (i) to pass a value to the **`$MT_Progress`** 1304 variable and (ii) to trigger the OnProgressUpdate 401 event of subpages that have been designed to show the progress of server actions.


Loop variable $loop
For Each 1 to 10
Progress Update
Value: $loop
Limit update frequency (ms):
If mt-progress-cancelling()=true()

The following are the key points:

- The *Value* setting is the value that will be passed to the **`$MT_Progress`** 1304 variable. In the screenshot above, the value that is passed will move from 1 to 10 for each iteration of the loop. In this way, the **`$MT_Progress`** 1304 variable will continuously hold the current status of the iterations through the loop. While iterating through a loop is straightforward, you must choose, for your set of actions, a suitable value to indicate the progress of the actions.
- Bear in mind that each time the Progress Update action is executed, the OnProgressUpdate 401 event of the subpage that will show the progress (see Progress Show Subpage 795) will be triggered. Each time this event is triggered the value of **`$MT_Progress`** 1304 will contain the current value supplied by the *Value* setting of the action.
- The *Limit Update Frequency* setting sets the number of milliseconds between updates that trigger the OnProgressUpdate 401 event of the subpage. If the action is expected to be executed with a high frequency, then this setting can specify that only those that occur at the specified frequency will trigger the OnProgressUpdate 401 event of the subpage. For example, if the action is expected to be executed five times a second (or once every 200 ms), then you can set this option to 1000 ms so that the update is sent only once every second.

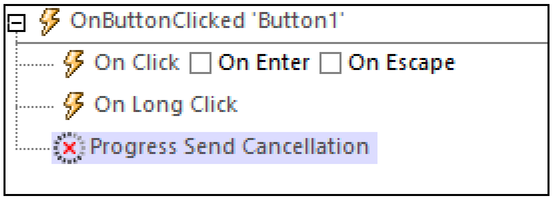
See the Progress Indicator tutorial 241 for an example of how to use this action.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

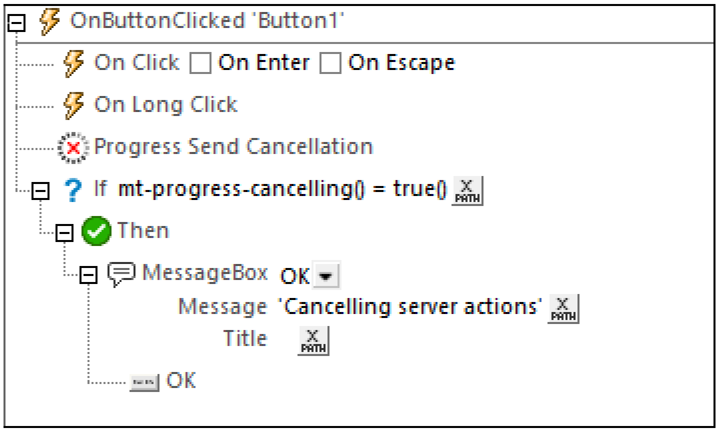
## 10.11.3 Progress Send Cancellation

The Progress Send Cancellation action sends a cancellation request from the client device. When this action is triggered on the client, the MobileTogether XPath extension function **`mt-progress-cancellation()`** 1262 is set to `true()`.


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Progress Send Cancellation

The **mt-progress-cancellation()** 1262 function can then be used as a test to determine whether to go ahead with actions to cancel server actions.

You can use the **mt-progress-cancelling()** 1262 function not only to run a cancellation procedure on the server, but also to run a cancellation procedure on the client (that is, on the subpage). For example, you might want to display a cancellation message for the user while the cancellation procedure is running on the server (*screenshot below*).


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Progress Send Cancellation
If mt-progress-cancelling() = true()
Then
MessageBox OK
Message 'Cancelling server actions'
Title
OK

See the Progress Indicator tutorial 241 for an example of how to use this action.

## MobileTogether extension functions

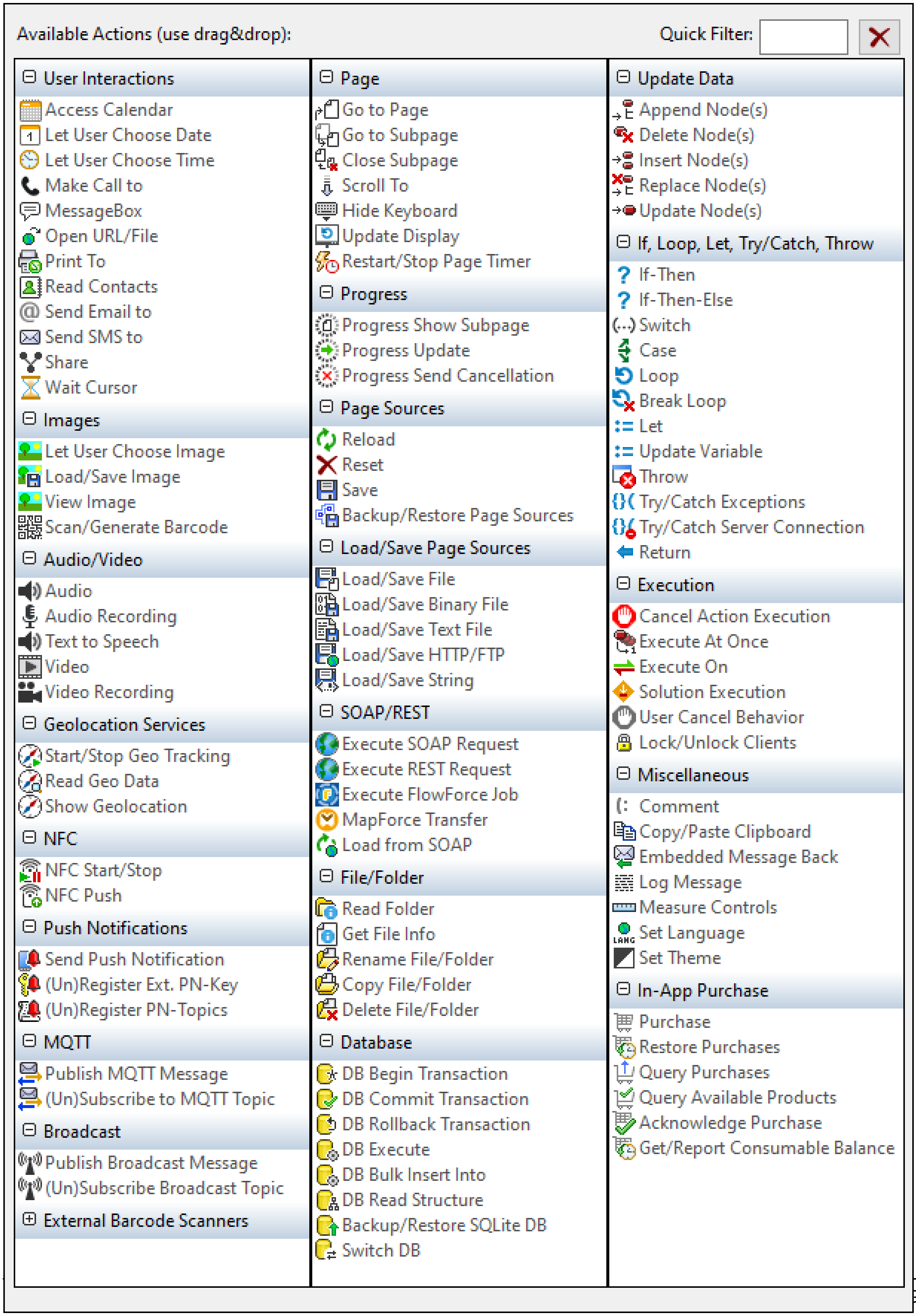
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**mt-progress-cancellation()**

# 10.12 Page Sources

The following actions are available in the Page Sources group of the Actions dialog (*screenshot below*):

- Reload 801
- Save 803
- Reset 802
- Backup/Restore Page Sources



Available Actions (use drag&drop):

Quick Filter:

⊟ User Interactions

- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images

- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video

- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services

- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC

- NFC Start/Stop
- NFC Push

⊟ Push Notifications

- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT

- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast

- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page

- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress

- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources

- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources

- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST

- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder

- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database

- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data

- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw

- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution

- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous

- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

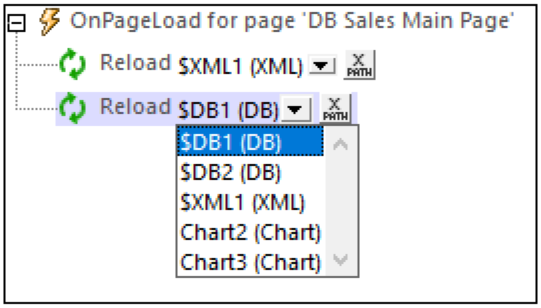
⊟ In-App Purchase

- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events* 389 *and* *Control Events* 665.

## 10.12.1 Reload

Reloads the external resource specified in the action's combo box (*see screenshot below*). External resources referenced by the project are available as entries in the combo box, and include XML files, charts, and images.



Note the following:

- If the resource to be updated is a page source, then an XPath expression that locates the root node of the page source can be used (for example: `$XML1` or `$DB2`).
- Notice that the screenshot above contains entries for chart controls. If selected for update, the chart image will be updated when the event is triggered. Image controls can also have their linked images updated with the Update 886 action.
- Updates can also be triggered by control events. This means, for example, that when a combo box is edited, the `OnFinishEditing` event of the combo box can be set to update the image linked to an image control. See the Quick Start (Part 1) tutorial 86 for an example of this.
- To reload multiple resources when the event is triggered, add multiple `Reload` actions, as has been done in the screenshot above.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

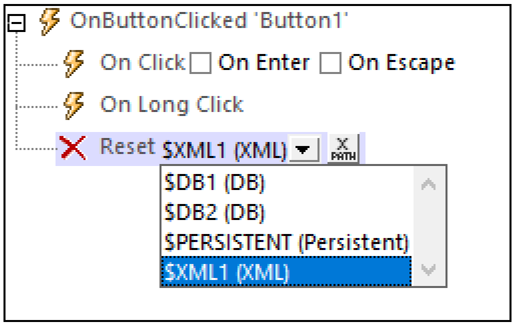
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

```
mt-external-error-code()
mt-get-source-from-name()
mt-get-source-name()
mt-get-source-structure()
```

## 10.12.2 Reset

Resets the page source that is selected in the action's combo box to the default values defined in the Page Sources Pane 270. The page source can also be entered via an XPath expression (for example: `$XML1`). Note that you can reset any kind of page source that has been defined in the Page Sources Pane 270, including the **`$PERSISTENT`** 349, **`$MT_GEOLOCATION`** 349, **`$MT_FILEINFO`** 349, and **`$MT_NFC`** 349 page source trees.



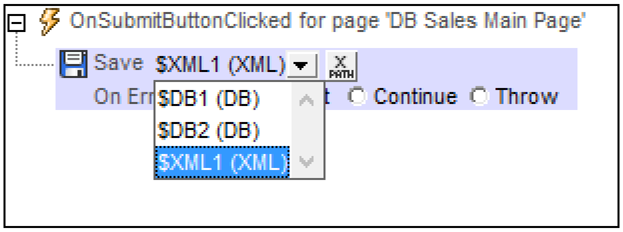
**Note:** If a variable locates a root node (**`$XML`**, **`$PERSISTENT`**, etc) and its infoset is changed because the Reset action is executed, then the variable is automatically updated to locate data in the new infoset. Note that this applies only when infoset modifications are caused by the Reset action (as opposed to, say, a data update caused during the execution of some other action).

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.12.3 Save

Saves data of the page source that is selected in the action's combo box to the default file of that page source. The data source must be an editable XML file or DB. To save data to multiple data sources, add multiple `Save` actions. Note that page sources that are read as JSON will also be saved as JSON (not as XML even though the data is presented in the GUI as an XML tree); *also see* *Page Source Options* 345.



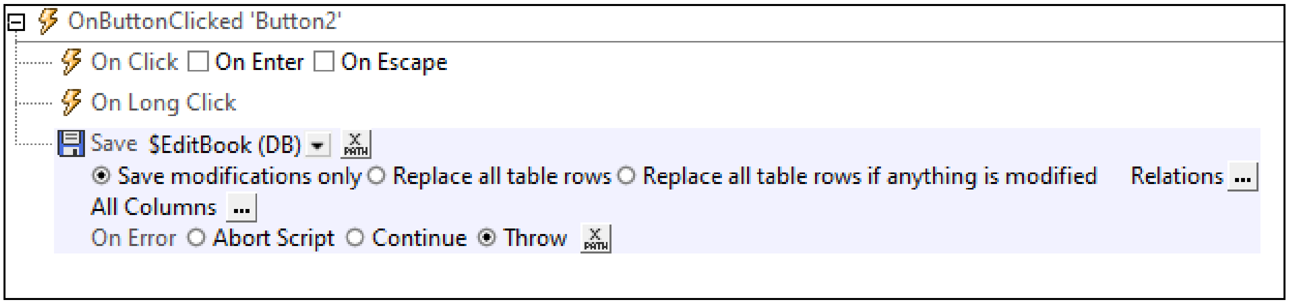
*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw*: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

### Save to DB

If the data source that is being saved is a DB, then, by default, all editable columns are selected for being updated (*see screenshot below*). You can choose to save the modified data only (primary key needed), or all table rows (no primary key needed). If you select *Replace All Table Rows*, then all rows in the DB are deleted and all the rows of the page source are inserted; note, however, that primary keys of new rows are not saved to the DB. There is also an option to save all table rows only if some data anywhere in the table has been modified. The *Save Modifications Only* option uses the primary key to check for modifications, and saves the modifications only; new rows are saved with their primary keys. If you wish to select an option that checks for modifications, then make sure that an **`OriginalRowSet`** 957 has been created for the table; otherwise an error will be reported on validation. To define how changes are to be saved to related tables, click the *Relations* button.



OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

On Long Click

Save $EditBook (DB)

● Save modifications only ○ Replace all table rows ○ Replace all table rows if anything is modified Relations ...

All Columns ...

On Error ○ Abort Script ○ Continue ● Throw

*Save related tables*

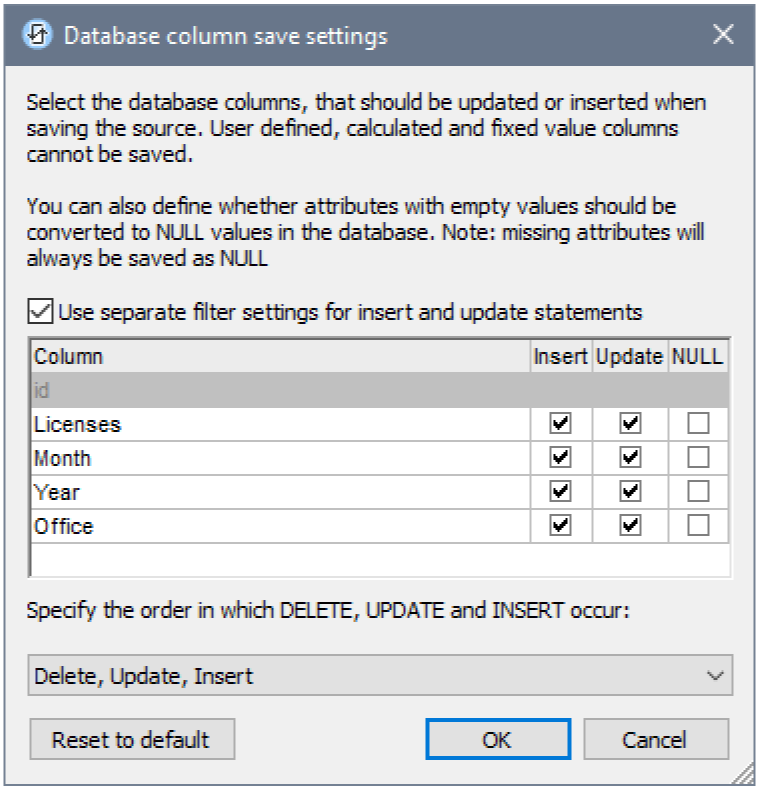
To specify how to save related tables, click the **Relations** button ... . This displays the Database Relation Save Settings dialog, which shows the related tables. In the combo box of each related table you can choose from the following options: (i) Replace all table rows of the related table; (ii) In the related table, save modifications only; (iii) Not save any changes to the related table.

When making these choices, take into account any private-key to foreign-key relations that exist between the main and related tables.

You can also access the save settings of the related table of a DB page source via the context menu of the page source[359] in the Page Sources Pane[270].

*Save DB columns*

To select specific columns to update, click the **All Columns** button ... . This displays the Database Column Save Settings dialog (*screenshot below*).



Database column save settings

Select the database columns, that should be updated or inserted when saving the source. User defined, calculated and fixed value columns cannot be saved.

You can also define whether attributes with empty values should be converted to NULL values in the database. Note: missing attributes will always be saved as NULL

☑ Use separate filter settings for insert and update statements

| Column | Insert | Update | NULL |
|---|---|---|---|
| id | | | |
| Licenses | ☑ | ☑ | □ |
| Month | ☑ | ☑ | □ |
| Year | ☑ | ☑ | □ |
| Office | ☑ | ☑ | □ |

Specify the order in which DELETE, UPDATE and INSERT occur:

Delete, Update, Insert

The dialog displays the columns of the DB page source. You can specify which columns can be updated or can take inserted values. (Updates refer to modified data in existing row elements; inserted values refer to data in newly added row elements.) By default, the *Insert* and *Update* options of each column are selected together as a pair. If, however, you wish to specify different options for a column's *Insert* and *Update* options, check the *Use separate filter settings for Insert and Update statements* check box. Attributes with empty values can be converted to `NULL` values in the DB by checking that column's *NULL* check box. Note that missing attributes will always be saved as `NULL`.

Columns that cannot be updated (because they are user-defined, fixed-value, or calculated-value) will not have an *Insert*, *Update*, or *NULL* option check box. In the screenshot above, the `ID` column cannot be updated because it holds fixed values. Deselect the columns you do not want to update.

You can specify the order in which deletions, updates, and insertions occur by selecting the desired order in the combo box at the bottom of the dialog.

If you wish to reset the *Save settings* so that all columns are updated, click **Reset to default**.

## MobileTogether extension functions

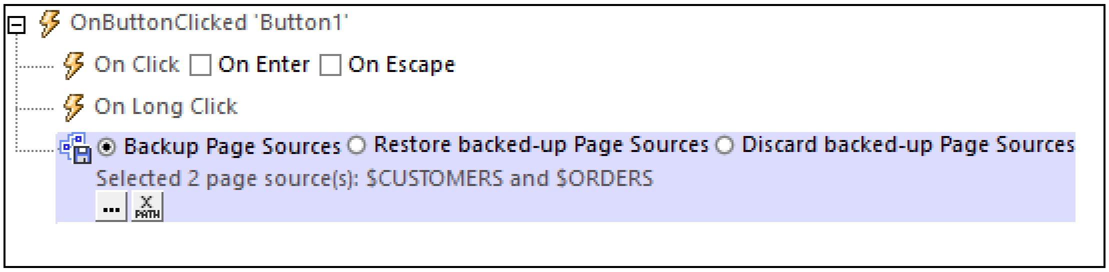
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-external-error-code()`**

**`mt-external-error-text()`**
**`mt-external-error-code()`**
**`mt-get-source-from-name()`**
**`mt-get-source-name()`**
**`mt-get-source-structure()`**

## 10.12.4 Backup/Restore Page Sources

Each Backup/Restore Page Sources action consists of three sub-actions (*see screenshot below*), of which one may be selected.



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

● Backup Page Sources ○ Restore backed-up Page Sources ○ Discard backed-up Page Sources

Selected 2 page source(s): $CUSTOMERS and $ORDERS

The three sub-actions are:

- *Backup Page Sources:* Backs up an internal copy of one or more page sources. Select the page source/s to backup by clicking the **Additional Dialog** button at the bottom of the action and selecting one or more of the available page sources.(*see screenshot above*).
- *Restore Backed-up Page Sources:* After an internal copy of a page source has been backed up (with *Backup Page Sources*), a page source might be further modified. The *Restore Backed-up Page Sources* action returns the page source to the state of the most recent internally saved copy.
- *Discard Backed-up Page Sources:* After an internal copy of a page source has been backed up (with *Backup Page Sources*) and that page source has been further modified, the *Discard Backed-up Page Sources* action discards the previously backed-up internal copy.

**Note:** The Backup/Restore Page Sources action applies only to page sources that have been backed up temporarily—that is, they are **not** saved to file. To save to file, use other actions like Save[803] or Save File[809].

*Usage*
The Backup/Restore Page Sources action enables you to back up a page source temporarily, and then to accept or discard further modifications on the basis of whether one or more conditions are met. For example, before going to a subpage, a page source can be backed up internally. After the user has edited data on the sub page, they can press buttons, respectively, to confirm or cancel the changes. The buttons would be defined, respectively, to discard and restore backed up page sources.

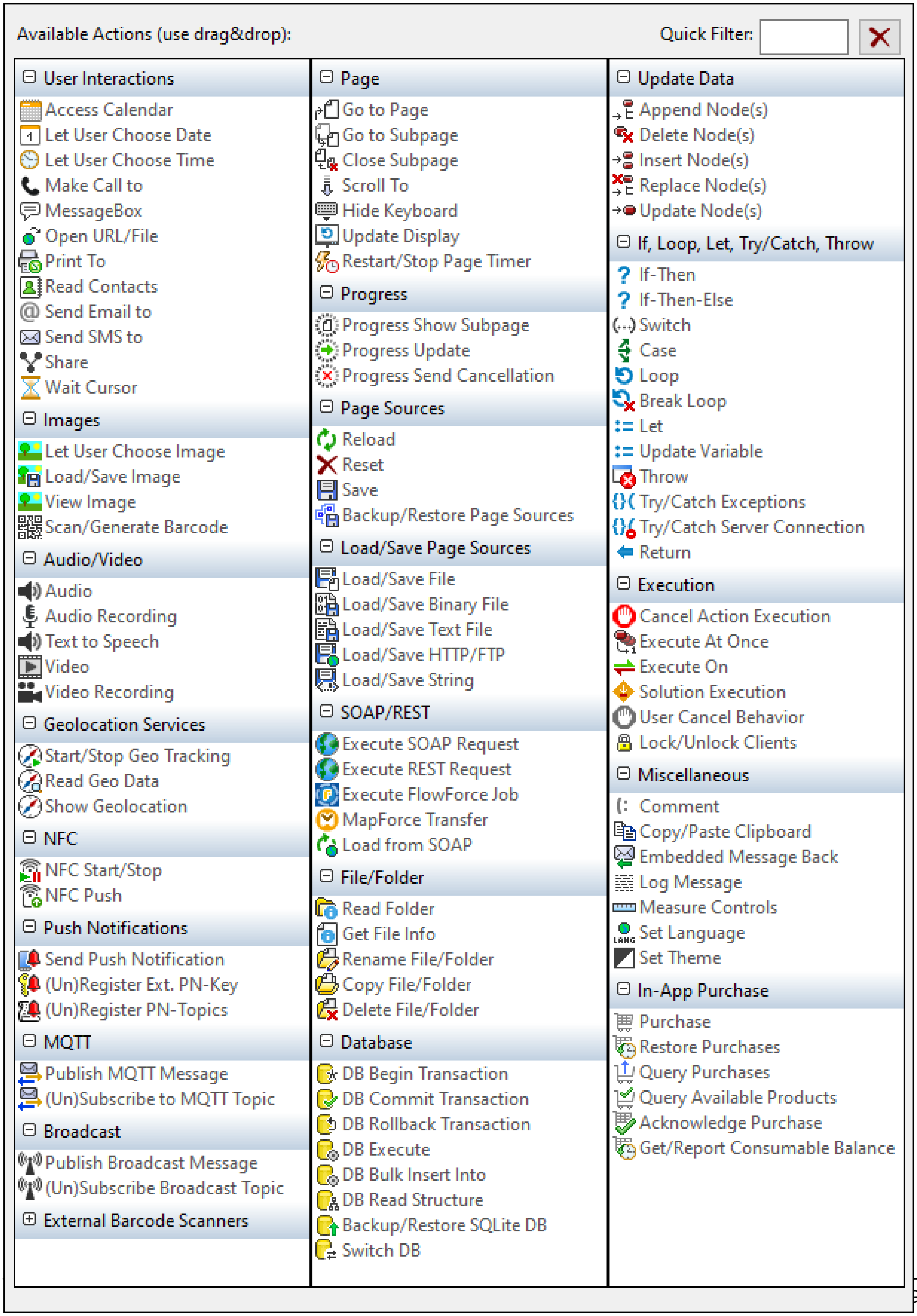
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.13 Load/Save Page Sources

The following actions are available in the Load/Save Page Sources group of the Actions dialog (*screenshot below*):

- Load/Save File 809
- Load/Save Binary File 815
- Load/Save Text File 821
- Load/Save HTTP/FTP 827
- Load/Save String



⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

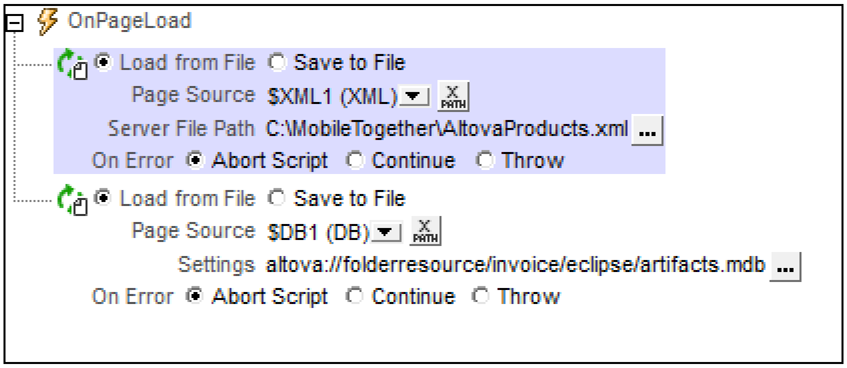
The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events*[389] *and* *Control Events*[665].

## 10.13.1 Load/Save File

You can set the action to either: (i) load data from a file, or (ii) save data to a file. To specify whether it is a load action or a save action that is carried out, select the appropriate radio button (*see screenshots below*).

### Load from file

For each `LoadFromFile` action, you can select one page source from the action's combo box (*see screenshot below*). Alternatively, you can use an XPath expression that locates the root node of the page source (for example: `$XML1`). You can then specify a file from which to load data for that page source when the event is triggered.



OnPageLoad

● Load from File ○ Save to File
Page Source $XML1 (XML)
Server File Path C:\MobileTogether\AltovaProducts.xml
On Error ● Abort Script ○ Continue ○ Throw

● Load from File ○ Save to File
Page Source $DB1 (DB)
Settings altova://folderresource/invoice/eclipse/artifacts.mdb
On Error ● Abort Script ○ Continue ○ Throw

To load data for multiple page sources when the event is triggered, add multiple `LoadFromFile` actions, as shown in the screenshot above.

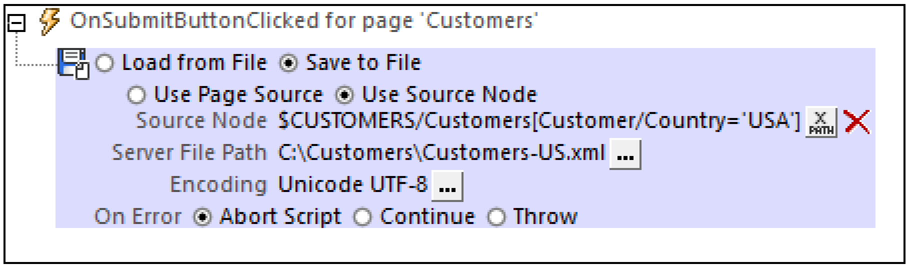
*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

## Save to file

Saves data from the selected page source or page source node (both are selected via the first XPath expression), to the XML file specified in the *File Path* field (*see screenshot below*). The encoding of the XML file is specified in the *Encoding* field. To save data for multiple data sources, add multiple `SaveToFile` actions. Note that you have the option of selecting either the entire page source (*Use Page Source*) or a subtree of a page source (*Use Source Node*).



OnSubmitButtonClicked for page 'Customers'

○ Load from File ● Save to File

○ Use Page Source ● Use Source Node

Source Node $CUSTOMERS/Customers[Customer/Country='USA']

Server File Path C:\Customers\Customers-US.xml

Encoding Unicode UTF-8

On Error ● Abort Script ○ Continue ○ Throw

**Note:** The `SaveToFile` action cannot be used to save to DBs. For saving to DBs, use the Save [803] action and select the DB page source you want to save to, or use the `DBExecute` [861] action.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

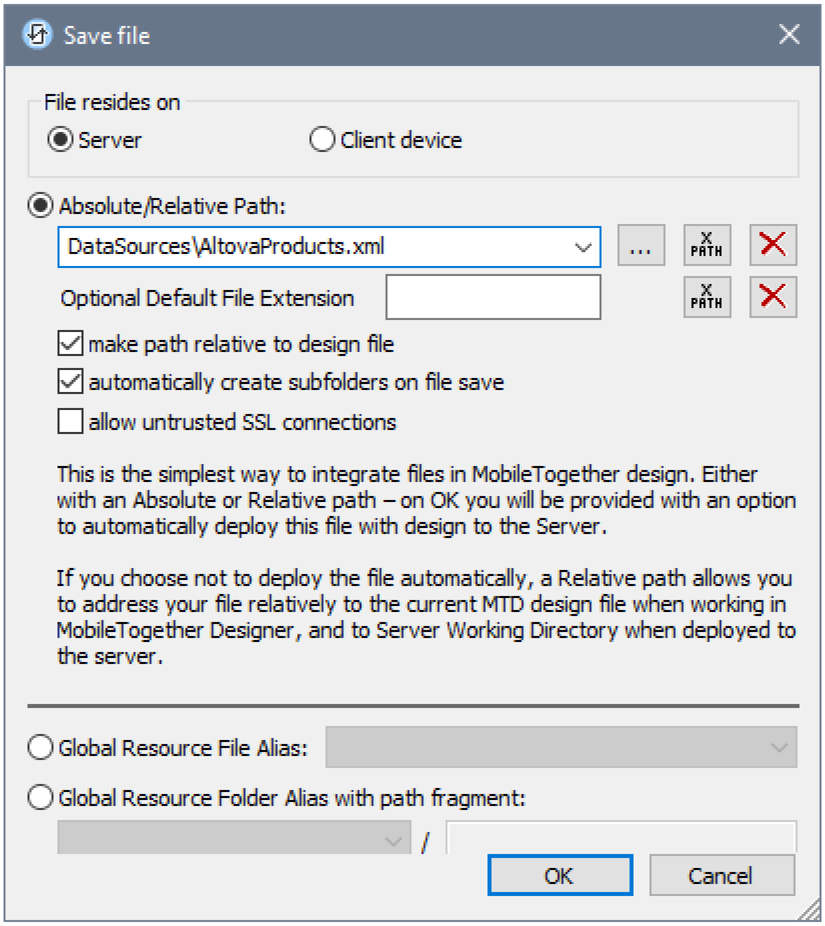
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable [907]. The Catch part of the Try/Catch action [907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action [907] for details.

## File locations

When you click the **Additional Dialog** button of the *File Path* field of the Load/Save File action (*see screenshots above*), the Save File dialog appears. In this dialog, you specify whether the file is located on the server or the client by selecting the respective radio button (*see screenshots below*).

*File is located on server*

If the file is located on the server, you can either browse for its location (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want.



Save file
File resides on
● Server
○ Client device
● Absolute/Relative Path:
DataSources\AltovaProducts.xml
...
XPATH
Optional Default File Extension
XPATH
☑ make path relative to design file
☑ automatically create subfolders on file save
□ allow untrusted SSL connections

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:
○ Global Resource Folder Alias with path fragment:
/
OK
Cancel

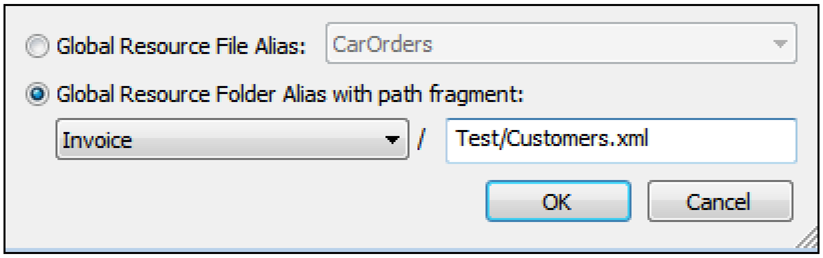
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). See the section Altova Global Resources 1334 for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

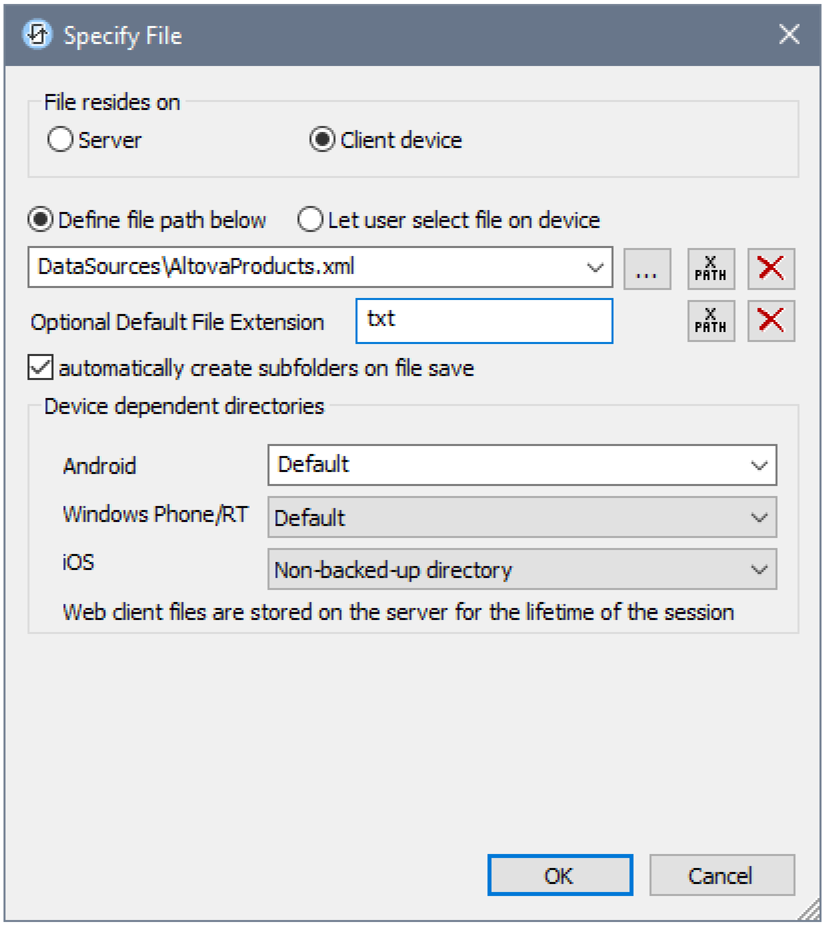
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration** 1654). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 1334 for details.

*File is located on client*

If the file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



Specify File

File resides on

○ Server ● Client device

● Define file path below ○ Let user select file on device

DataSources\AltovaProducts.xml

Optional Default File Extension: txt

☑ automatically create subfolders on file save

Device dependent directories

| | |
|---|---|
| Android | Default |
| Windows Phone/RT | Default |
| iOS | Non-backed-up directory |

Web client files are stored on the server for the lifetime of the session

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To[686] (*Source File* and *Target File* options), Load/Save File[809], Load/Save Image[707], Load/Save Binary File[815], Load/Save Text File[821], Read Folder[847], and Get File Info[849].

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.
- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the

directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog[1663] (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).

- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is

no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.

- *Throw*: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-last-file path()`**
**`mt-external-error-code()`**
**`mt-extract-file-extension()`**
**`mt-extract-file-name()`**
**`mt-external-error-code()`**
**`mt-get-source-from-name()`**
**`mt-get-source-name()`**
**`mt-get-source-structure()`**
**`mt-save-json-to-string()`**

# 10.13.2 Load/Save Binary File

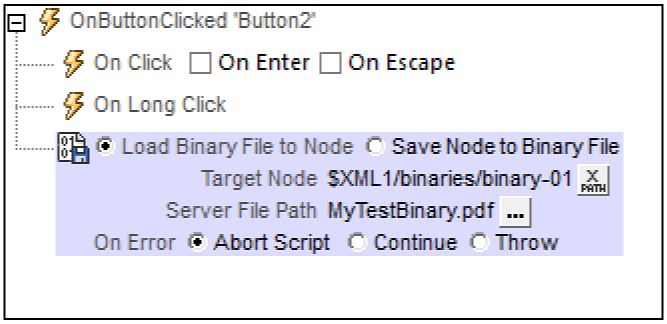
The Load/Save Binary File action enables the following:

- Loading a binary file into a page source node as Base64-encoded content
- Saving the Base64-encoded content of a page source node as a binary file to the client or to a server-side location.

This allows binary files, such as PDFs, to be held as XML content.

## Loading a binary file into a page source node

A binary file can be loaded into a page source node by using the *Load Binary to Node* option of the Load/Save Binary File action (*see screenshot below*). Use an XPath expression to select the target node, that is, the page source node where the binary data will be stored. In the *File Path* field, select the binary file that is to be loaded into the target node. The binary file data is converted to Base64 and stored as Base64-encoded content in the target node.



OnButtonClicked "Button2"

On Click ☐ On Enter ☐ On Escape

On Long Click

● Load Binary File to Node ○ Save Node to Binary File

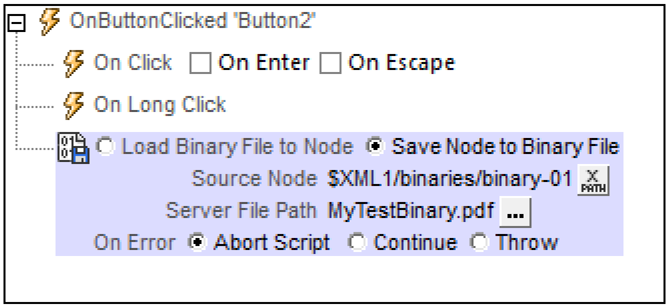
Target Node $XML1/binaries/binary-01

Server File Path MyTestBinary.pdf

On Error ● Abort Script ○ Continue ○ Throw

## Saving Base64-encoded content as a binary file

Base64-encoded content that is stored in a page source node can be saved as a binary file by using the *Save Binary to File* option of the Load/Save Binary File action (*see screenshot below*). Select the page source node where the Base64-encoded content is located (the *Source Node* field; *see screenshot below*). Then select the location on the server or client where the file is to be saved (the *File Path* field).



OnButtonClicked "Button2"

On Click ☐ On Enter ☐ On Escape

On Long Click

○ Load Binary File to Node ● Save Node to Binary File

Source Node $XML1/binaries/binary-01

Server File Path MyTestBinary.pdf

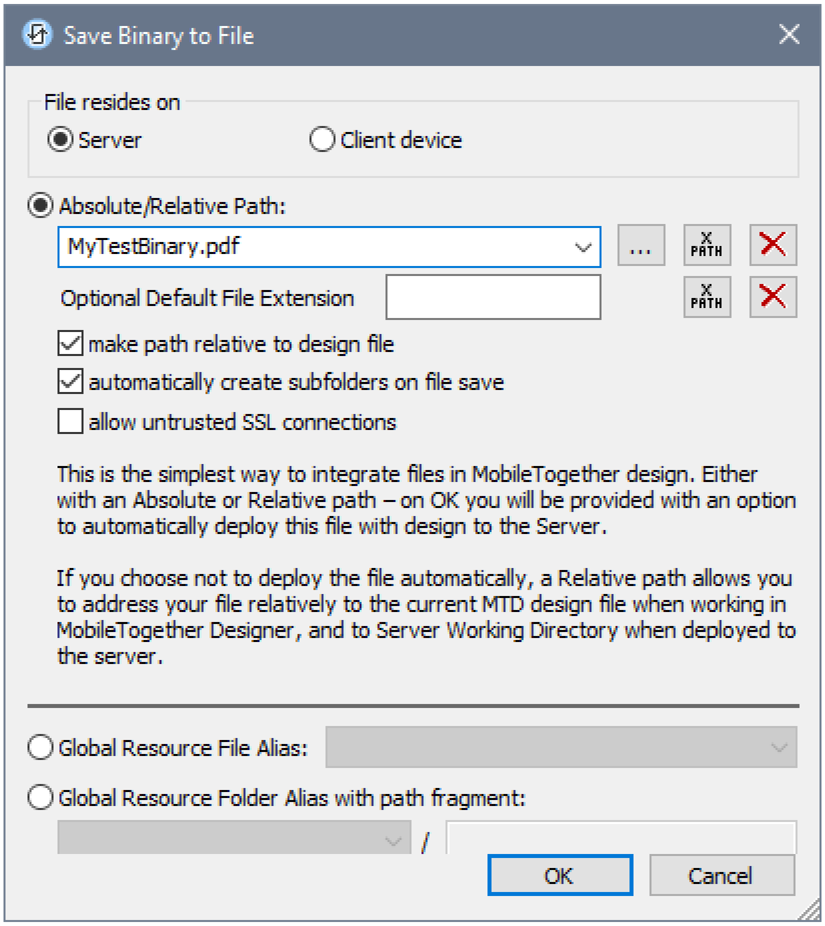
On Error ● Abort Script ○ Continue ○ Throw

## Binary file locations

When you click the **Additional Dialog** button of the *File Path* field of the Load/Save Image action (*see screenshots above*), the Save Binary to File dialog appears. In this dialog, you specify whether the file is located on the server or the client by selecting the respective radio button (*see screenshots below*).

*<u>File is located on server</u>*

If the binary file is located on the server, you can either browse for its location (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want.



Save Binary to File
File resides on
● Server
○ Client device
● Absolute/Relative Path:
MyTestBinary.pdf
...
XPATH
Optional Default File Extension
XPATH
☑ make path relative to design file
☑ automatically create subfolders on file save
□ allow untrusted SSL connections

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:
○ Global Resource Folder Alias with path fragment:
/
OK
Cancel

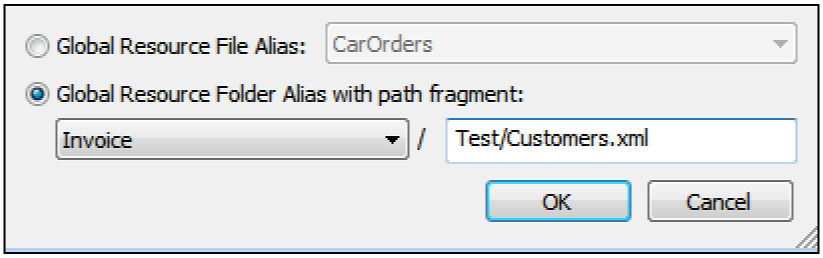
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). See the section Altova Global Resources[1334] for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

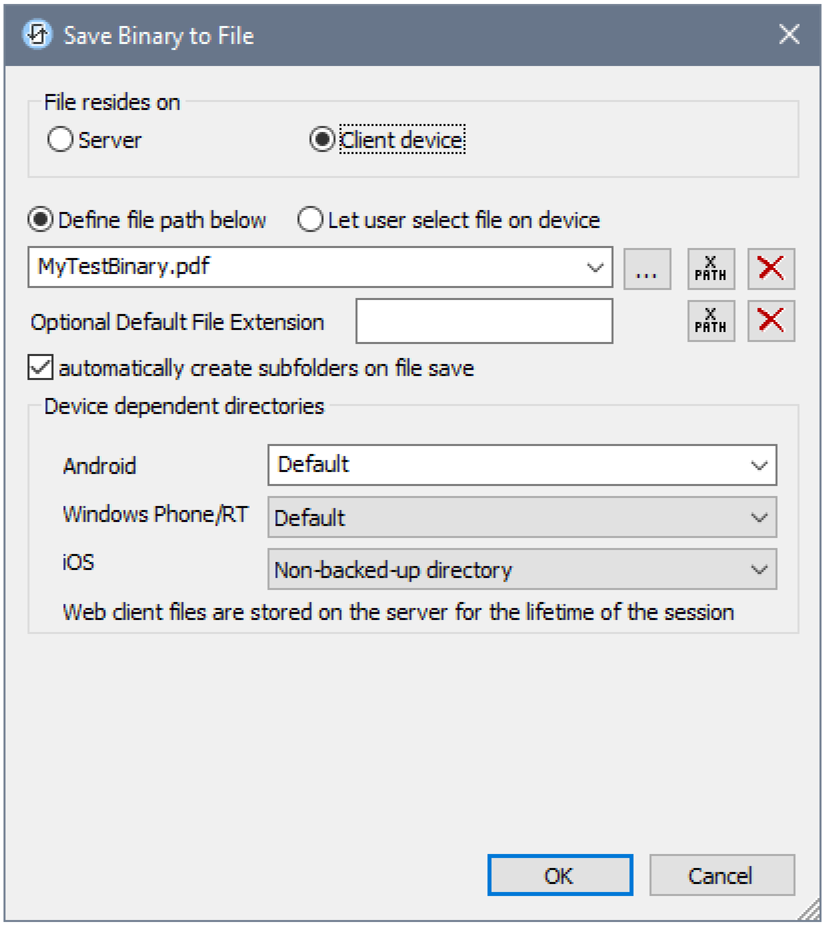
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources[1334] for details.

*File is located on client*

If the binary file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.


Save Binary to File
File resides on
○ Server
● Client device
● Define file path below
○ Let user select file on device
MyTestBinary.pdf
...
Optional Default File Extension
☑ automatically create subfolders on file save
Device dependent directories
Android
Default
Windows Phone/RT
Default
iOS
Non-backed-up directory
Web client files are stored on the server for the lifetime of the session
OK
Cancel

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To 686 (*Source File* and *Target File* options), Load/Save File 809, Load/Save Image 707, Load/Save Binary File 815, Load/Save Text File 821, Read Folder 847, and Get File Info 849.

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.
- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.
- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the

directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog 1663 (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).

- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is

no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.

- *Throw*: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

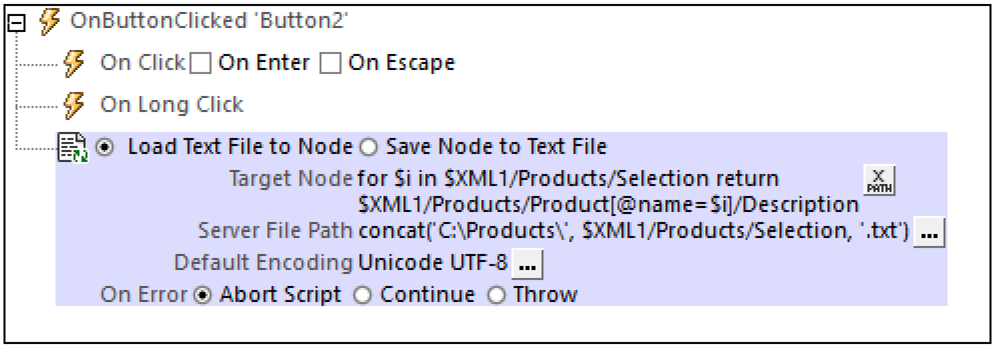
```
mt-last-file path()
mt-external-error-code()
mt-external-error-text()
mt-external-error-code()
mt-get-source-from-name()
mt-get-source-name()
mt-get-source-structure()
```

# 10.13.3 Load/Save Text File

You can set the action to either: (i) load data from a text file to a page source node, or (ii) save data from a page source node to a text file. To specify whether it is a load action or a save action that is carried out, select the appropriate radio button (*see screenshots below*).

## Load from text file

For each `LoadTextFile` action, you can select one target node from a page source. For example, in the screenshot below, the `Description` node is the target node. Next, specify a text file from which to load text data into the target node. You can also select a default encoding. If the encoding of the text cannot be found automatically, then the default encoding will be used.



⊟ OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

On Long Click

● Load Text File to Node ○ Save Node to Text File

Target Node for $i in $XML1/Products/Selection return $XML1/Products/Product[@name=$i]/Description

Server File Path concat('C:\Products\', $XML1/Products/Selection, '.txt') ...

Default Encoding Unicode UTF-8 ...

On Error ● Abort Script ○ Continue ○ Throw

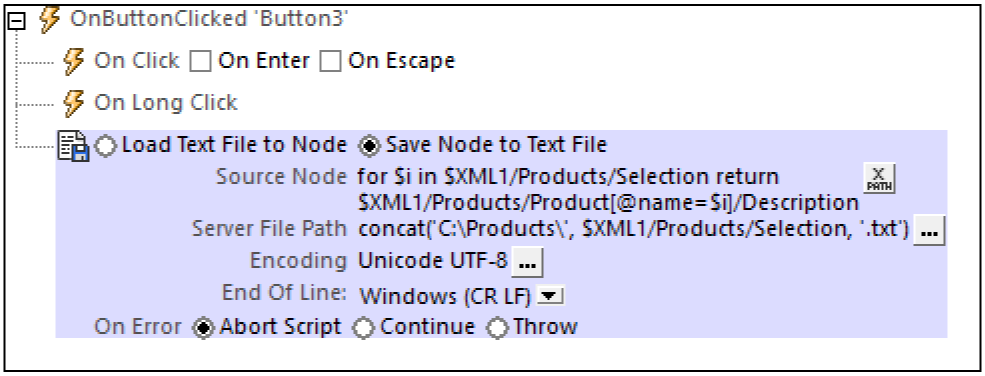
To load data for multiple target nodes when the event is triggered, add multiple `LoadTextile` actions.

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

## Save to text file

Saves data from the selected page source node to the text file specified in the *File Path* field (*see screenshot below*). The encoding and end-of-line characters of the text file are specified, respectively, in the *Encoding* and *End of Line* fields.


OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
○ Load Text File to Node ● Save Node to Text File
Source Node: for $i in $XML1/Products/Selection return $XML1/Products/Product[@name=$i]/Description
Server File Path: concat('C:\Products\', $XML1/Products/Selection, '.txt')
Encoding: Unicode UTF-8
End Of Line: Windows (CR LF)
On Error ● Abort Script ○ Continue ○ Throw

To save data for multiple page sources, add multiple `SaveNode` actions.

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for

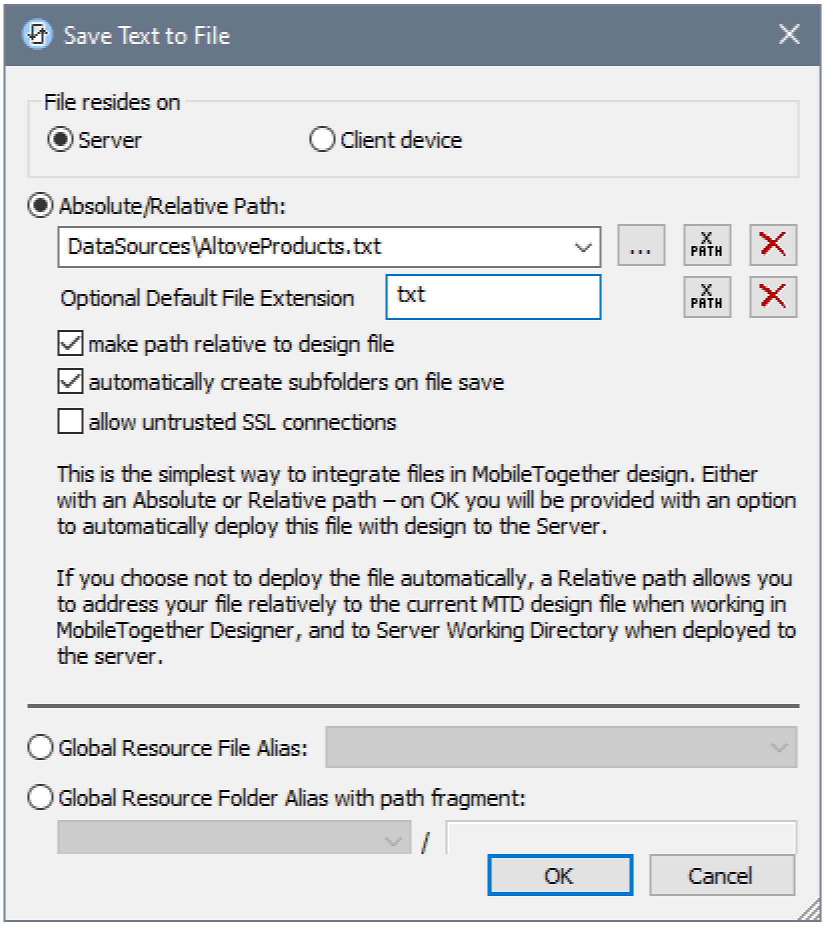
details.

## File locations

When you click the **Additional Dialog** button of the *File Path* field of the Load/Save Text File action (*see screenshots above*), the Save Text to File dialog appears. In this dialog, you specify whether the file is located on the server or the client by selecting the respective radio button (*see screenshots below*).

*File is located on server*

If the file is located on the server, you can either browse for its location (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want.


Save Text to File

File resides on

● Server ○ Client device

● Absolute/Relative Path:

DataSources\AltoveProducts.txt

Optional Default File Extension: txt

☑ make path relative to design file

☑ automatically create subfolders on file save

□ allow untrusted SSL connections

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias:

○ Global Resource Folder Alias with path fragment:

- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section Location of Project Files 289 for details. You can also choose whether to allow untrusted

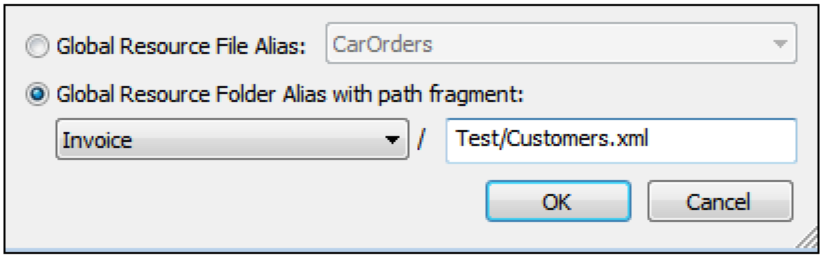
SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- *Allow untrusted SSL connections:* A certificate associated with a URL is considered untrusted if it isn't signed by a trusted root certificate or if it can't link to a trusted root certificate. If the certificate is signed by a major certificate authority, it just means that one of the chain certificates in between yours and the root is not installed on the web server. If a trusted certificate is expected (for example, because the HTTPS protocol is specified), then selecting this option enables connections also with URLs that have an untrusted certificate.

- *Global Resource File Alias:* Select a file alias from the file aliases available in the combo box. The available file aliases will be those currently defined in the Global Resources Definitions file. Each file alias maps to different file resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). See the section Altova Global Resources[1334] for details.

- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



○ Global Resource File Alias: CarOrders

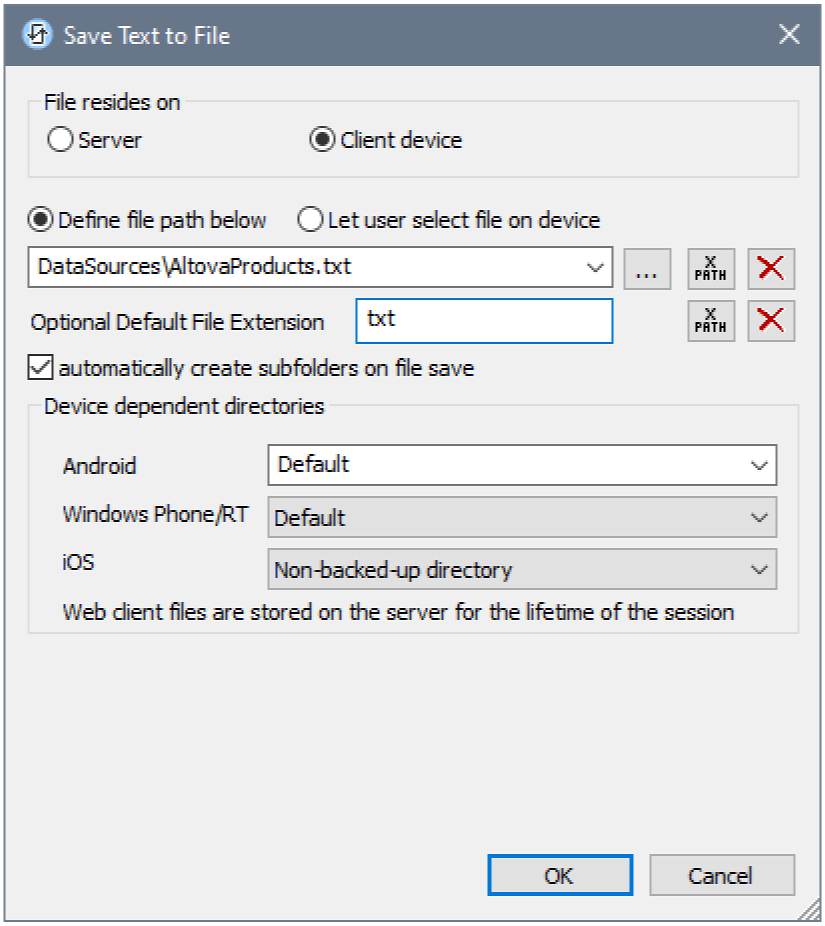
● Global Resource Folder Alias with path fragment:

Invoice / Test/Customers.xml

The available folder aliases will be those currently defined in the Global Resources Definitions file. Each folder alias maps to different folder resources according to the currently active configuration in MobileTogether Designer (selected via the command **Tools | Active Configuration**[1654]). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources[1334] for details.

*File is located on client*

If the file is located on the client, specify the path to it by entering/selecting the location, or by constructing the path with an XPath expression. Use the **Reset** button to remove the current entry.



Save Text to File

File resides on

○ Server ● Client device

● Define file path below ○ Let user select file on device

DataSources\AltovaProducts.txt

Optional Default File Extension: txt

☑ automatically create subfolders on file save

Device dependent directories

Android: Default

Windows Phone/RT: Default

iOS: Non-backed-up directory

Web client files are stored on the server for the lifetime of the session

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To 686 (*Source File* and *Target File* options), Load/Save File 809, Load/Save Image 707, Load/Save Binary File 815, Load/Save Text File 821, Read Folder 847, and Get File Info 849.

**Note:** Files on the client can also be saved to an SD card on the mobile device.

*Filename is defined below (by the designer of the solution)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

- *Device dependent directories:* Select the device directory from the dropdown list. On Windows Phone/RT and iOS, the allowed directories are pre-determined. On Android devices, in addition to the

directories in the dropdown list of the *Android* combo box, you can enter any folder you like. On Android and Windows Phone/RT, if you select *Default*, which is the default selection, the MobileTogether app's sandbox directory is selected. On iOS devices, MobileTogether creates two directories: (i) a *Backed-up directory* for files that are saved to the iCloud, and which can then be re-downloaded; (ii) a *Non-backed-up directory* for files that do not need to be backed up. Select *Backed-up directory* or *Non-backed-up directory* as required. In web browsers, files are located relative to the browser's sandbox.

- *File locations for simulations:* Since files located on the client will not be available during simulations, you can specify a folder that will stand in for the client folder during simulations. Files within this stand-in folder must, of course, have the same names as the files specified in the design. This folder is specified in the Simulation tab of the Options dialog 1663 (**Tools | Options**).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab *(see the MobileTogether Server user manual)*.

*Filename is defined by the end user (on the client device)*

- *Default file extension for file saving:* When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- *Optional File Filter:* The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt','html,'xml'`).

- *Optional Default File:* You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- *Web Message Box:* Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- *Automatically create subfolders on file save:* If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is

no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.

- *Throw*: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

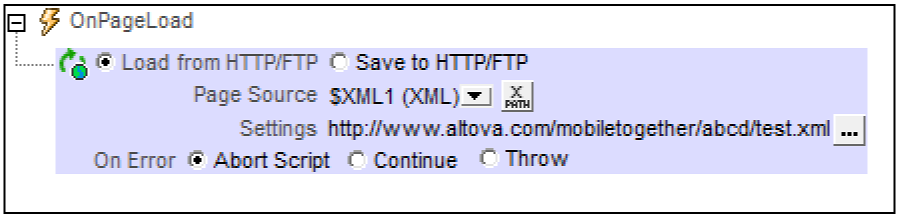
```
mt-last-file path()
mt-external-error-code()
mt-external-error-text()
mt-external-error-code()
mt-get-source-from-name()
mt-get-source-name()
mt-get-source-structure()
mt-load-json-from-string()
mt-load-string()
mt-save-json-to-string()
```

# 10.13.4 Load/Save HTTP/FTP

You can set the action to either: (i) load data from an HTTP/FTP file, or (ii) save data to a file via HTTP/FTP. To specify whether it is a load action or a save action that is carried out, select the appropriate radio button (*see screenshot below*).

## Load from HTTP/FTP

For each `LoadFromHTTP/FTP` action, you can select one page source from the available page sources and specify an HTTP/FTP source from which to load data. (Alternatively, you can use an XPath expression that locates the root node of the page source (for example: `$XML1`).) When the event is triggered, data from the HTTP/FTP source will be loaded into the page source you have specified. To load data for multiple page sources, add multiple `LoadFromHTTP/FTP` actions.



OnPageLoad

● Load from HTTP/FTP ○ Save to HTTP/FTP

Page Source $XML1 (XML)

Settings http://www.altova.com/mobiletogether/abcd/test.xml

On Error ● Abort Script ○ Continue ○ Throw

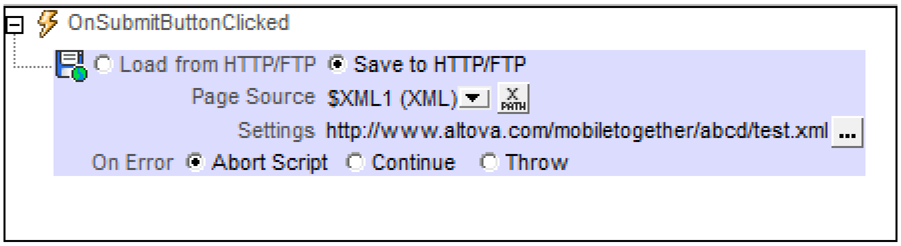
*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

## Save to HTTP/FTP

Saves the page source that is selected in the action's combo box to an XML or HTML file at a target HTTP or FTP location that is specified in the *Settings* field of the action's definition (*see screenshot below*). (Alternatively, you can use an XPath expression that locates the root node of the page source (for example: `$XML1`).) To enter access details of the HTTP/FTP location, click ... . This displays the Edit Web Access Settings dialog[317] for selecting HTTP/FTP sources; here you can enter the file's URL and security settings.



OnSubmitButtonClicked

○ Load from HTTP/FTP ● Save to HTTP/FTP

Page Source $XML1 (XML)

Settings http://www.altova.com/mobiletogether/abcd/test.xml ...

On Error ● Abort Script ○ Continue ○ Throw

To save data from multiple page sources or to multiple destinations, add multiple `SaveToHTTP/FTP` actions. To add another `SaveToHTTP/FTP` action, drag the `Load/SaveHTTP/FTP` action into the event tab, and set its radio button to the `SaveToHTTP/FTP` action.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.13.5 Load/Save String

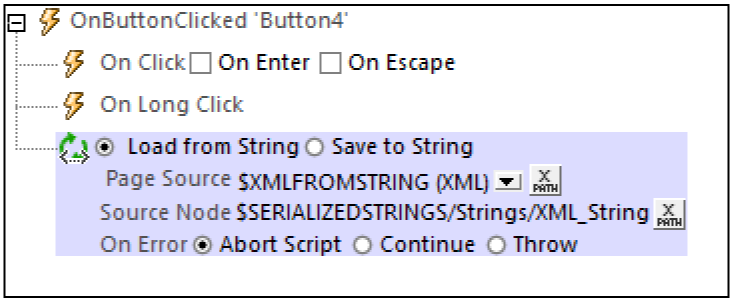
The two actions, **Load from String** and **Save to String**, respectively:

- load JSON or XML data from a string into a JSON/XML page source, and
- serialize a JSON/XML page source to a string, and save the serialized string to a location specified by an XPath expression.

These functions are particularly useful in Embedded Webpage Solutions[1427], where serialized XML data can be received in a message from a webpage and stored in the **`$MT_EMBEDDEDMESSAGE`** JSON page source. The Load from String action can take the XML string and generate an XML page source. Conversely, an XML page source can be serialized to a string with the Save to String action and stored in a page source node.

### Load from String

The Load from String action parses the string that is targeted by the XPath expression of the *Source Node* setting (*see screenshot below*), and generates the structure and data of the selected page source (*see screenshot*). Select the page source from the Page Source combo box, or an XPath expression that locates the root node of the page source (for example: `$XML1`).



OnButtonClicked 'Button4'
On Click ☐ On Enter ☐ On Escape
On Long Click
● Load from String ○ Save to String
Page Source $XMLFROMSTRING (XML)
Source Node $SERIALIZEDSTRINGS/Strings/XML_String
On Error ● Abort Script ○ Continue ○ Throw

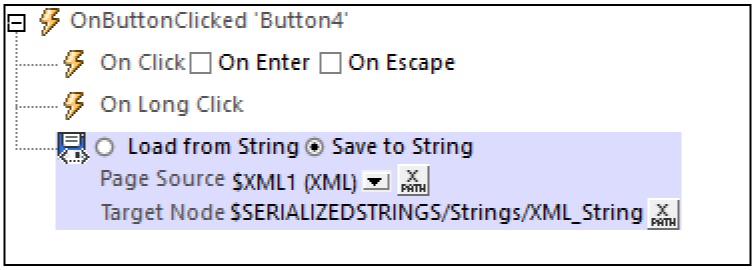
The type of the page source selected (JSON or XML) should correspond to the serialization of the string. So, if the source node string is a JSON string, then a JSON document will be generated. If the source node string is an XML string, then this string is loaded into the page source, which must be an XML page source; the root element in the serialized string will be created as the root element of the XML page source.

- *About the source string:* In the screenshot above, an XML string is located in the source node **`$SERIALIZEDSTRINGS/Strings/XML_String`**. This is the string that will be loaded into the selected page source. In order for this to work correctly, the page source must be an XML page source. (The XPath expression of the *Source Node* setting need not take a string from a node; the string can also be entered directly in the XPath expression.)

- *Encoding of the source string:* The following encodings are supported on clients, so the source string must be in one of these encodings: UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32BE, US-ASCII, ISO 8859-1.
- *About the JSON/XML page source:* The page source needs to have been created before it is selected in the action. The JSON/XML document that is loaded from the string is created as the entire page source within the document node. If, at run time, the structure of the loaded document does not match the structure of the page source as defined in the design, then the solution will not be executed correctly. This is because the design works with the node names of an expected page source structure, but the nodes of the page source that is actually created have different names.

## Save to String

The Save to String action (*screenshot below*) serializes the page source named in the *Page Source* option, and saves the serialized string to the location that is specified by the XPath expression of the *Target Node* setting. You can also select the page source via the setting's combo box or via an XPath expression that locates the root node of the page source (for example: `$XML1`).



OnButtonClicked 'Button4'

On Click □ On Enter □ On Escape

On Long Click

○ Load from String ● Save to String

Page Source $XML1 (XML)

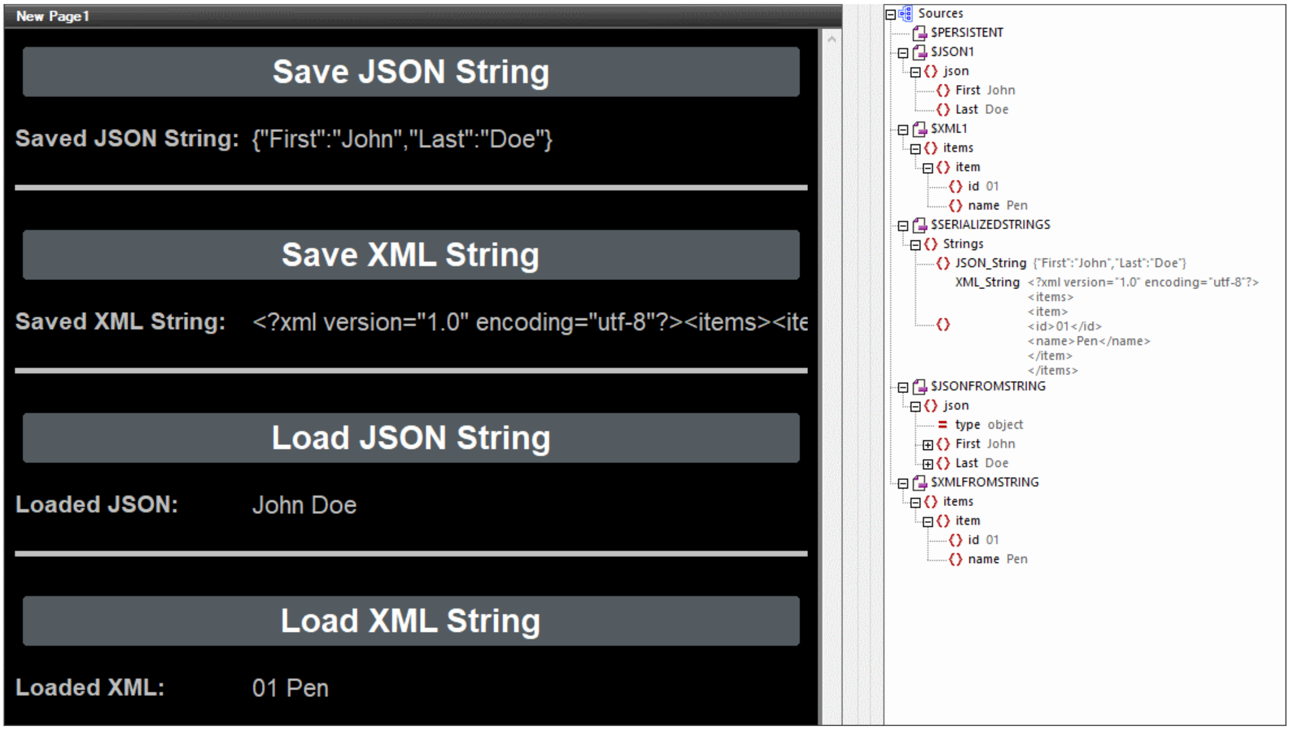
Target Node $SERIALIZEDSTRINGS/Strings/XML_String

Note the following points:

- The page source and its structure must exist at design time so that it can be selected as a combo box option.
- The content of the entire page source, no matter of what type, will be serialized, from its first character to its last character, to a string.
- The string that results from the serialization will be saved to the node specified in the XPath expression.

## Example file

An example file named **`LoadSaveString.mtd`** demonstrates the usage of these two actions. This file is located in the *(My) Documents* folder: `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\Actions`. This example contains two page sources (**`$JSON1`** and **`$XML1`**), with each containing, respectively, fixed JSON and XML data (*see screenshot*). By using the Save to String 829 action, each page source can be serialized as a string and saved to a node. We then reverse this action by using the Load from String 829 action to load the just-serialized strings into new page sources (**`$JSONFROMSTRING`** and **`$XMLFROMSTRING`**).



Saved JSON String: {"First":"John","Last":"Doe"}

Saved XML String: <?xml version="1.0" encoding="utf-8"?><items><ite

Loaded JSON: John Doe

Loaded XML: 01 Pen

```
Sources
  $PERSISTENT
  $JSON1
    json
      First John
      Last Doe
  $XML1
    items
      item
        id 01
        name Pen
  $SERIALIZEDSTRINGS
    Strings
      JSON_String {"First":"John","Last":"Doe"}
      XML_String <?xml version="1.0" encoding="utf-8"?>
                 <items>
                 <item>
                 <id>01</id>
                 <name>Pen</name>
                 </item>
                 </items>
  $JSONFROMSTRING
    json
      type object
      First John
      Last Doe
  $XMLFROMSTRING
    items
      item
        id 01
        name Pen
```

- The design file contains a JSON page source (**`$JSON1`**) and XML page source (**`$XML1`**), the structures and data of which are defined in the file itself (*see screenshot of simulation above*). Both data structures are rudimentary.
- The design contains a button to respectively save each page source as a serialized string (**Save JSON/XML String**), each in a different node of another page source named **`$SERIALIZEDSTRINGS`**. Both these buttons use the Save to String[829] action.
- The generated serializations are used to create new pages sources, respectively, **`$JSONFROMSTRING`** and **`$XMLFROMSTRING`**. This is done by triggering the Load from String[829] actions of the third and fourth buttons (**Load JSON/XML String**).

This example, in effect, uses the two actions (Save to String[829] and Load from String[829]) to make a round trip from one page source to another page source via a serialized string. The origin and destination page sources will have the same structure and data.

## MobileTogether extension functions

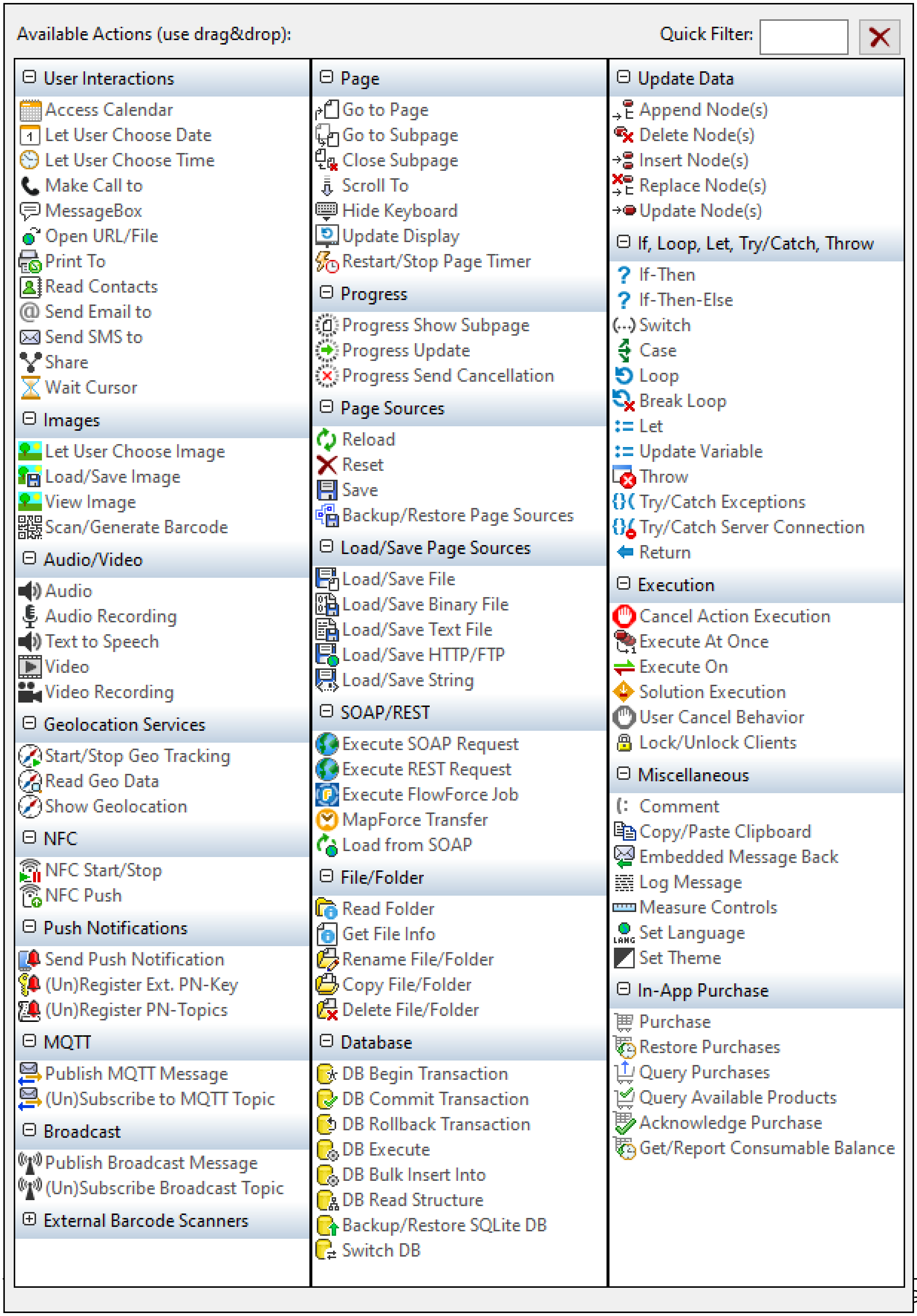
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-load-json-from-string()`**
**`mt-load-string()`**
**`mt-save-json-to-string()`**
**`mt-string-to-hexBinary()`**

# 10.14 SOAP/REST

The following actions are available in the SOAP/REST group of the Actions dialog (*screenshot below*):

- Execute SOAP Request 835
- Execute REST Request 837
- Execute FlowForce Job 838
- MapForce Transfer 839
- Load from SOAP



Available Actions (use drag&drop):

Quick Filter:

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

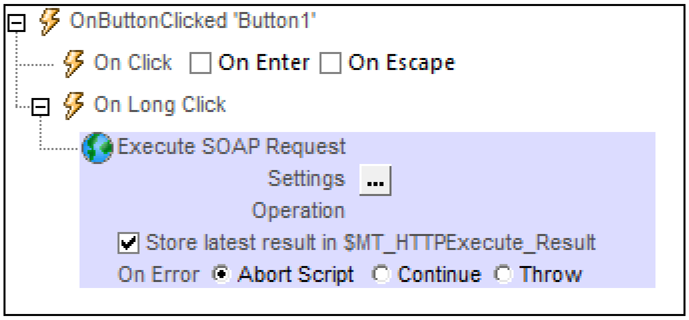
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events* 389 *and Control Events* 665.

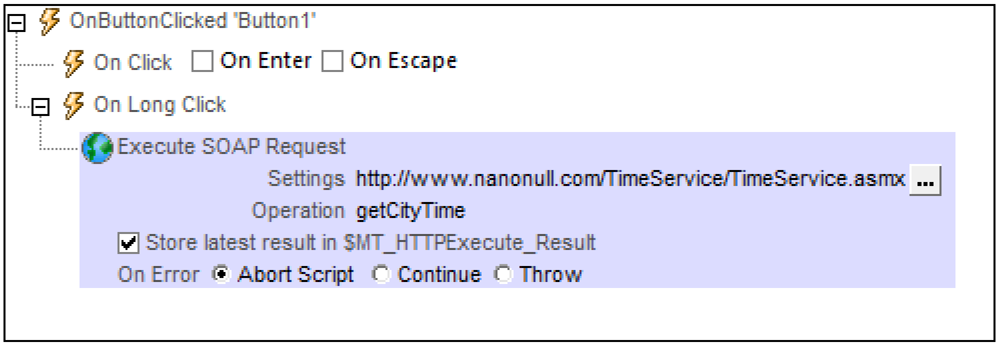
## 10.14.1 Execute SOAP Request

Executes a SOAP request that is generated from a WSDL file. To choose the WSDL file, click the **Additional Settings** button of the *Settings* field (*see screenshot below*).



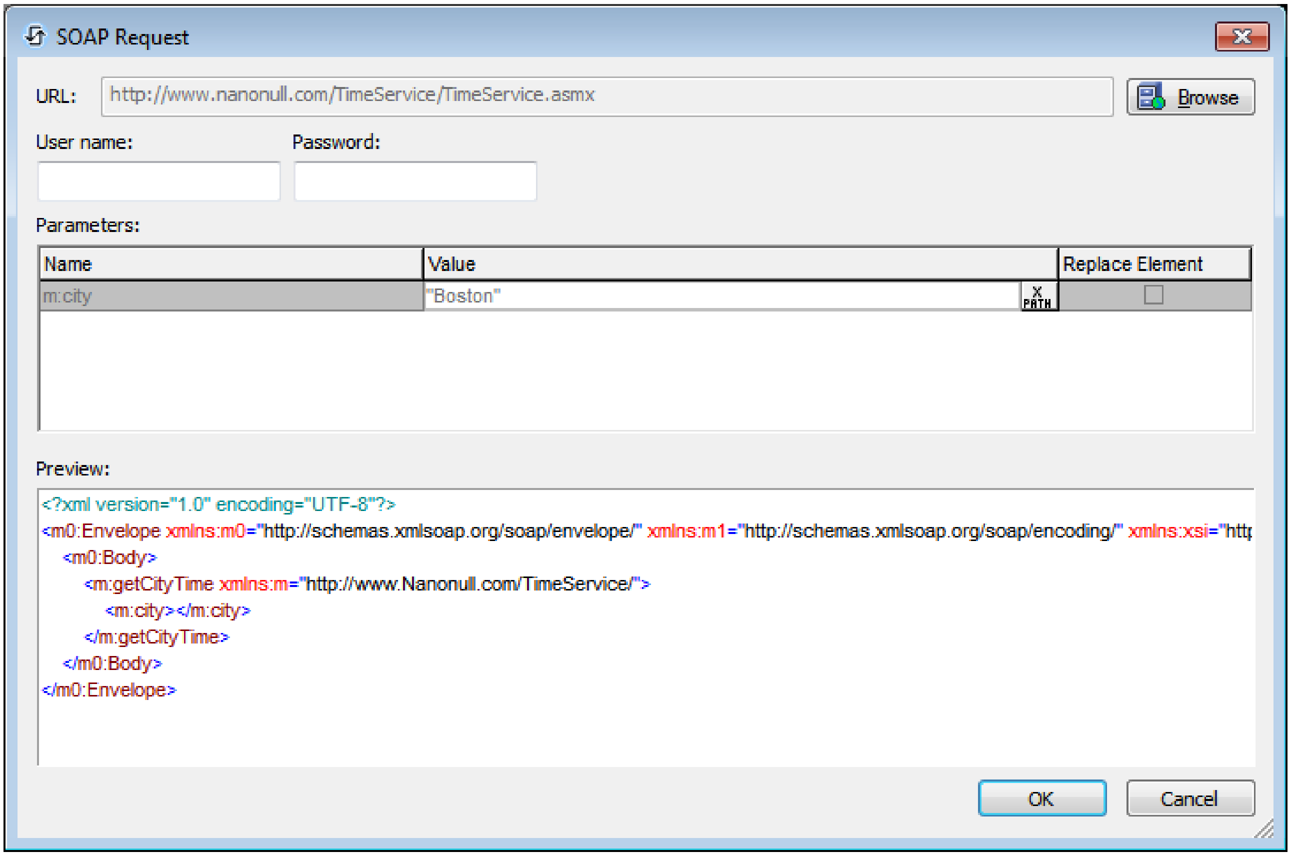
OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Execute SOAP Request
Settings ...
Operation
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw

Choose your WSDL file, and select the SOAP operation you want. The SOAP request is automatically generated from the WSDL file and is displayed in SOAP Request dialog. Click **OK** in the SOAP Request dialog to save this as the SOAP request to execute. The action now displays the URL of the web service to which the SOAP request will be sent at runtime (*see the Settings field in the screenshot below*). If you wish to store the response to the SOAP request, then check the *Store latest result* option (*see screenshot below*). The SOAP response will be saved in the `$MT_HTTPExecute_Result` 1304 variable. You can then use this variable to access the data in the SOAP response at some other location in the design. Note, however, that the `$MT_HTTPExecute_Result` 1304 variable can also be used by the Execute REST Request 837 and Execute FlowForce Job 838 actions. So the variable will contain the last result generated by *any* of the actions that use it.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx ...
Operation getCityTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw

If you wish to change the SOAP request after one has already been defined, click the **Additional Dialog** button of the *Settings* field (*see screenshot above*). This displays the SOAP Request dialog (*screenshot below*).



URL: http://www.nanonull.com/TimeService/TimeService.asmx

User name:

Password:

Parameters:

| Name | Value | Replace Element |
|---|---|---|
| m:city | "Boston" | □ |

Preview:

```
<?xml version="1.0" encoding="UTF-8"?>
<m0:Envelope xmlns:m0="http://schemas.xmlsoap.org/soap/envelope/" xmlns:m1="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http
  <m0:Body>
    <m:getCityTime xmlns:m="http://www.Nanonull.com/TimeService/">
      <m:city></m:city>
    </m:getCityTime>
  </m0:Body>
</m0:Envelope>
```

Click the **Browse** button of the *URL* field to choose a WSDL file and restart the process of defining the SOAP request that you want to execute.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

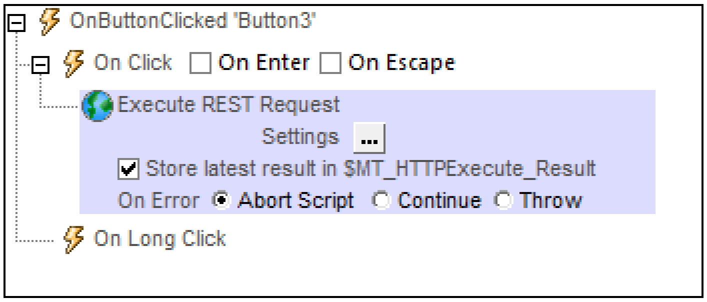
The tutorial SOAP Requests 215 shows how to use the Execute SOAP Request action.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.14.2 Execute REST Request

Executes a REST request that you define in the RESTful API Request dialog[328]. To open this dialog, click the **Additional Settings** button of the *Settings* field (*see screenshot below*).



OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
Execute REST Request
Settings ...
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw
On Long Click

After you have defined the REST request, the URL of the request is displayed in the *Setting*s field of the action. If you want to put or post content with your request, you can choose to send the content in the request body or as HTTP multi-part content. Content can be sent as a file, as an XPath that resolves to the content, or as Base64. For example, you can send an Image as a file or as Base64 content.

The request will be executed at runtime. If you wish to store the result of the request in the `$MT_HTTPExecute_Result`[1304] variable, check the *Store latest result* option (*see screenshot above*). You can then use the `$MT_HTTPExecute_Result`[1304] variable to access the result elsewhere in the design. Note, however, that this variable can also be used by the Execute SOAP Request[835] and Execute FlowForce Job[838] actions. So the variable will contain the last result generated by *any* of the actions that use it.

If you wish to change the REST request after one has already been defined, click the **Additional Dialog** button of the *Settings* field (*see screenshot above*). This displays the RESTful API Request dialog[328], in which you you can define the new request.

Error processing

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a

message box saying whether a page load was successful or not.

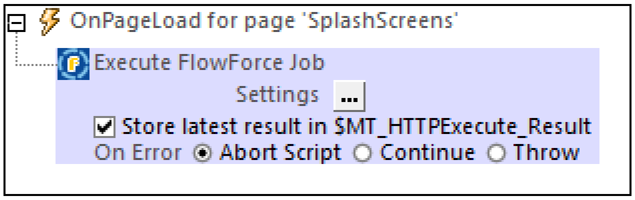
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.14.3 Execute FlowForce Job

Executes a FlowForce job request that you define in the Edit FlowForce Settings dialog[342]. To open this dialog, click the **Additional Settings** button of the *Settings* field (*see screenshot below*).



OnPageLoad for page 'SplashScreens'
Execute FlowForce Job
Settings ...
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw

After you have defined the FlowForce job request, the URL of the job is displayed in the *Settings* field of the action. The FlowForce job will be executed at runtime, and the result will be returned. If you wish to store the result of the request in the `$MT_HTTPExecute_Result`[1304] variable, check the *Store latest result* option (*see screenshot above*). You can then use the `$MT_HTTPExecute_Result`[1304] variable to access the result elsewhere in the design. Note, however, that this variable can also be used by the Execute SOAP Request[835] and the Execute REST Request[837] actions. So the variable will contain the last result generated by *any* of the actions that use it.

If you wish to change the FlowForce job request after one has already been defined, click the **Additional Dialog** button of the *Settings* field (*see screenshot above*). This displays the Edit FlowForce Settings dialog[342], in which you can define the new request.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.

- *Throw*: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.14.4 MapForce Transfer

The MapForce Transfer action enables one set of data structures to be converted (mapped) to a second set of data structures. Each data structure of the output set can be written to a file or to a node of a page source.

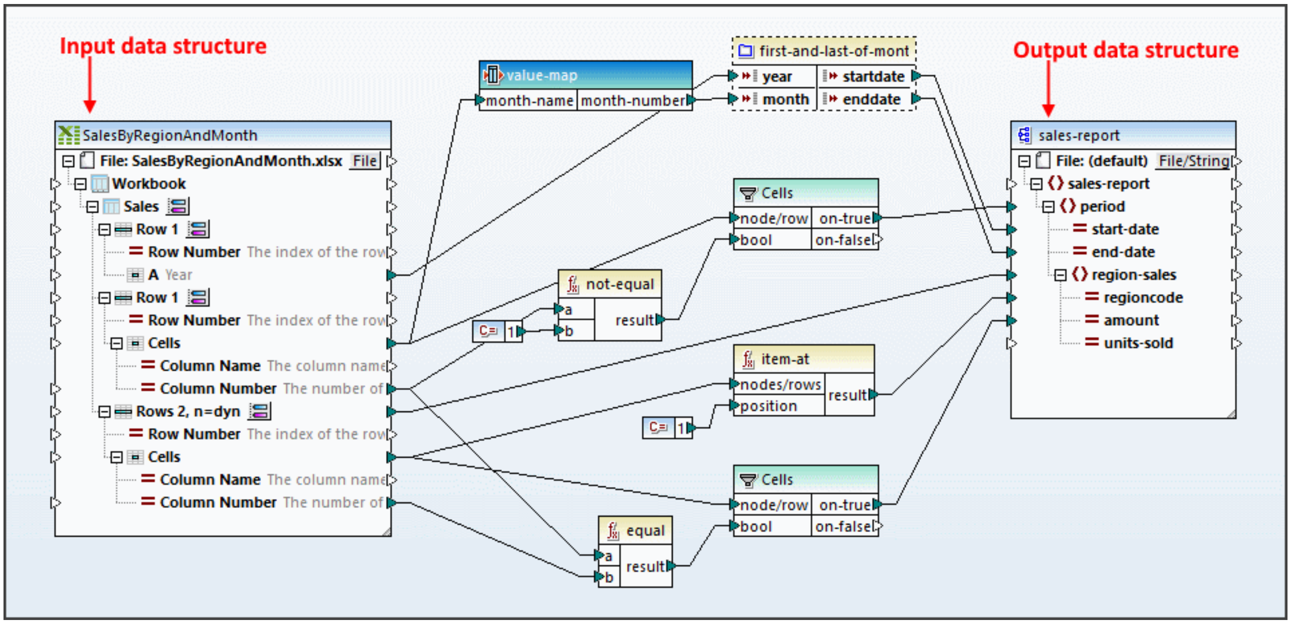
To carry out the MapForce transfer, you will need to use the following additional components:

- *Altova MapForce* to design the mapping of one set of input data structures to a set of output data structures. After the mapping has been designed, it is generated as a MapForce Server Execution (MFX or `.mfx`) file.
- *Altova MapForce Server* must be located on the same machine as MobileTogether Server. It is called by MobileTogether to process the MFX file and generate the output data structures.

**Note:** If you want to test the MapForce Transfer action in a local simulation 1356 or a trial run on client 1370, then MapForce Server must be installed on the same machine as MobileTogether Designer.

### A MapForce design

A MapForce design is created in Altova MapForce; it maps input data structures to one or more output data structures. In the example design shown below, there is one input data structure, which is an Excel file, and one output data structure, which is an XML file. Each data structure (or component in MapForce jargon) has a name. A design can have multiple input components and multiple output components.



After the mapping design has been completed, use MapForce's **File | Compile to MapForce Server Execution File** command to generate an MFX file. At runtime, the MFX file is used by MapForce Server to generate the output data structure.

In order to use the MapForce Transfer action of MobileTogether, you will need to know the following details of the MapForce design:

- The names of the input and output components that you want to use. (In the design shown above, these are, respectively, **`SalesByRegionAndMonth`** and **`sales-report`**.) Note that you can specify multiple input and output components. If, in the design, multiple input components are required to generate one output data structure, then they must all be supplied as (input and output) parameters to the MapForce Transfer action.
- In some MapForce designs, value parameters are used as part of the process of generating the mapped output data structure. Each such parameter takes a value, which is used in some way or the other towards generating the output data structure. You will need to know the names of any such parameters that you might need to correctly generate the output data structure.

For more information about creating MapForce designs and how mapping works, see the MapForce user manual. Also see the MapForce Server user manual.

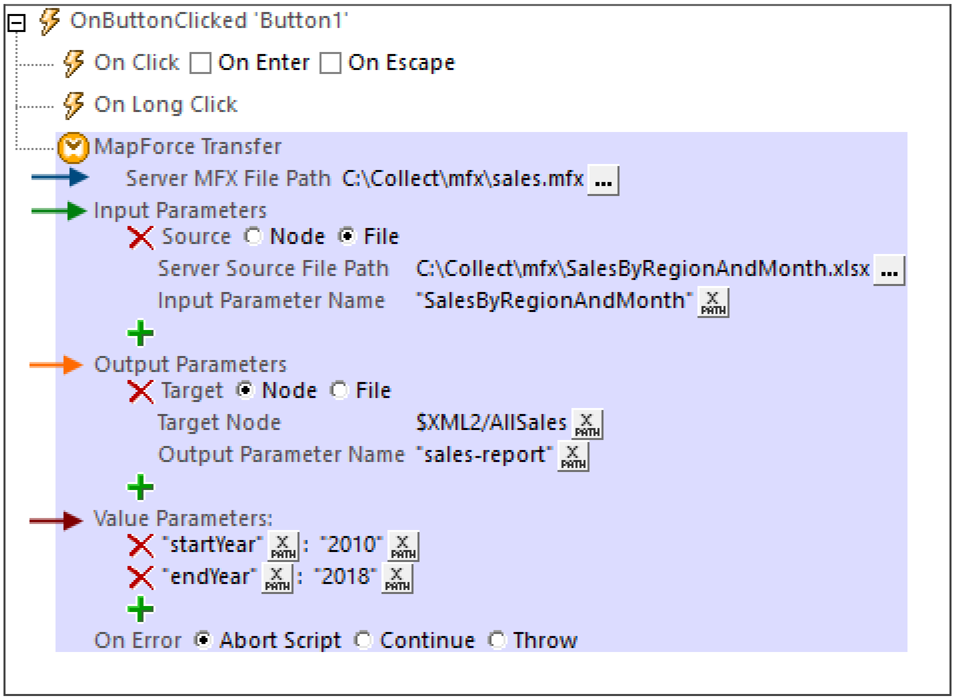
## MobileTogether's MapForce Transfer action

The MapForce Transfer action makes a call to MapForce Server, supplying it with an MFX file to process and the required input data. The screenshot below shows a MapForce Transfer action that is based on the mapping design shown above.

The action's settings are as follows:

- *Server MFX File Path* locates the MFX file to use.
- *Input Parameters*: Each input parameter specifies an input component of the mapping together with the data that should be used for this input component. You can specify multiple input parameters; make

sure that you specify all the input components that are required to correctly generate the output data structure you want. The *Input Parameter Name* setting specifies the name of the input component (from the MapForce design). You can then choose whether to supply the data for that input component from a page source node or from a file. Then select the node or file. The input component specified in the screenshot below is **`SalesByRegionAndMonth`**; since this component is an Excel data structure, an Excel file is supplied that has exactly the same structure as that of the input component. This is necessary for the mapping to be successful.



```
⊟ OnButtonClicked 'Button1'
    On Click □ On Enter □ On Escape
    On Long Click
    MapForce Transfer
        Server MFX File Path  C:\Collect\mfx\sales.mfx ...
    Input Parameters
        ✕ Source ○ Node ● File
          Server Source File Path   C:\Collect\mfx\SalesByRegionAndMonth.xlsx ...
          Input Parameter Name      "SalesByRegionAndMonth"
        +
    Output Parameters
        ✕ Target ● Node ○ File
          Target Node               $XML2/AllSales
          Output Parameter Name     "sales-report"
        +
    Value Parameters:
        ✕ "startYear": "2010"
        ✕ "endYear": "2018"
        +
    On Error ● Abort Script ○ Continue ○ Throw
```

- *Output Parameters*: Each output parameter specifies an output component of the mapping together with the location to which the output data structure should be written. You can specify multiple output parameters. The *Output Parameter Name* setting specifies the name of the output component (in the MapForce design) that you want to generate. You can then choose whether the output data structure should be written to a page source node or to a file. Then select the node or file. The output component specified in the screenshot above is **`sales-report`**, and it will be saved to the root element (**`AllSales`**) of the **`$XML2`** page source. Note that the entire output data structure will be copied to the node specified. So if an XML fragment is generated, then the entire XML fragment is copied to the specified node. You should make sure that the generated data structure will fit correctly into the structure of the target node.
- *Value Parameters*: These are *name–value* pairs that provide values as inputs for use in the mapping. The *name* part of a pair is the name of an input component in MapForce; the *value* part is the value to be passed to this component at run time. You can create as many value parameters as you like.

**Note:** At run time, the parameter settings you enter for this action are sent as the parameters of a call to MapForce Server. Therefore, each parameter name (across the combined set of input, output, and value parameters) must be unique.

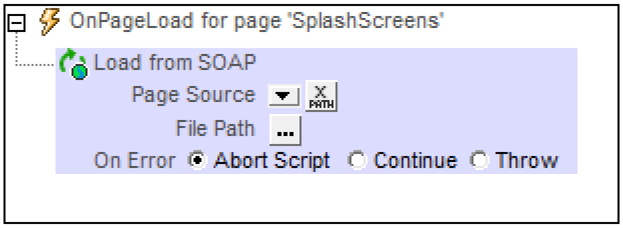
**Note:** MapForce Server must be installed on the same machine as MobileTogether Server.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

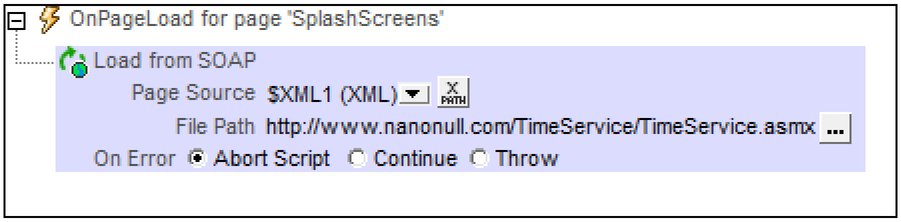
## 10.14.5 Load from SOAP

For each `LoadFromSOAP` action, you can load data from a SOAP request that is generated from a WSDL file. To choose the WSDL file, click the **Additional Settings** button of the *File Path* field (*see screenshot below*). The data is loaded into the selected page source.



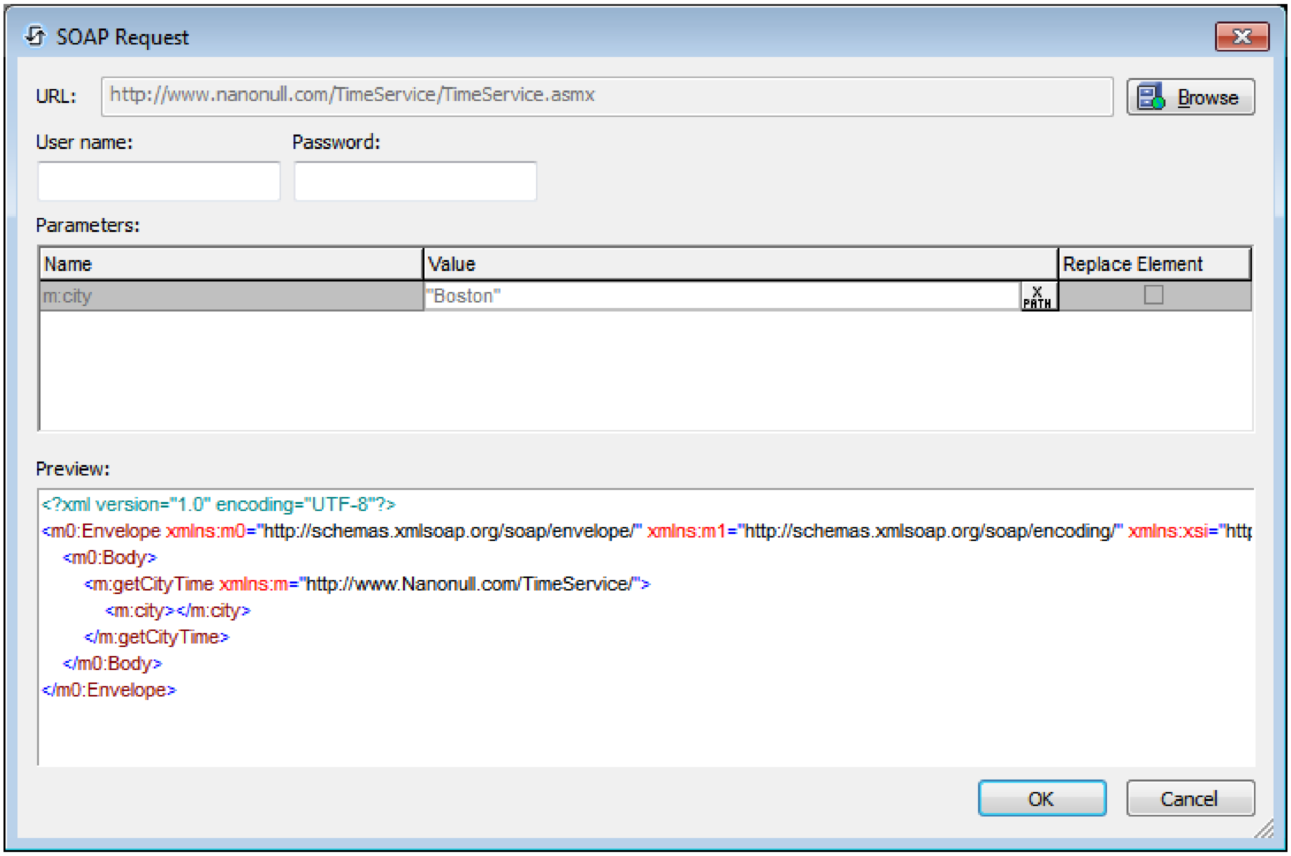
OnPageLoad for page 'SplashScreens'
Load from SOAP
Page Source
File Path
On Error ● Abort Script ○ Continue ○ Throw

Choose your WSDL file, and select the SOAP operation you want. The SOAP request is automatically generated from the WSDL file and is displayed in the SOAP Request dialog. Click **OK** in the SOAP Request dialog to save this as the SOAP request to use. The action now displays the URL of the web service to which the SOAP request will be sent at runtime (*see the Source field in the screenshot below*).



OnPageLoad for page 'SplashScreens'
Load from SOAP
Page Source $XML1 (XML)
File Path http://www.nanonull.com/TimeService/TimeService.asmx
On Error ● Abort Script ○ Continue ○ Throw

If you wish to change the SOAP request after one has already been defined, click the **Additional Dialog** button of the *File Path* field (*see screenshot above*). This displays the SOAP Request dialog (*screenshot below*). Click the **Browse** button of the *URL* field to choose a WSDL file and restart the process of defining the SOAP request.



URL: http://www.nanonull.com/TimeService/TimeService.asmx

| Name | Value | Replace Element |
|---|---|---|
| m:city | "Boston" | □ |

```
<?xml version="1.0" encoding="UTF-8"?>
<m0:Envelope xmlns:m0="http://schemas.xmlsoap.org/soap/envelope/" xmlns:m1="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http
   <m0:Body>
      <m:getCityTime xmlns:m="http://www.Nanonull.com/TimeService/">
         <m:city></m:city>
      </m:getCityTime>
   </m0:Body>
</m0:Envelope>
```

To load data from multiple data sources when the event is triggered, add multiple `LoadFromSOAP` actions.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

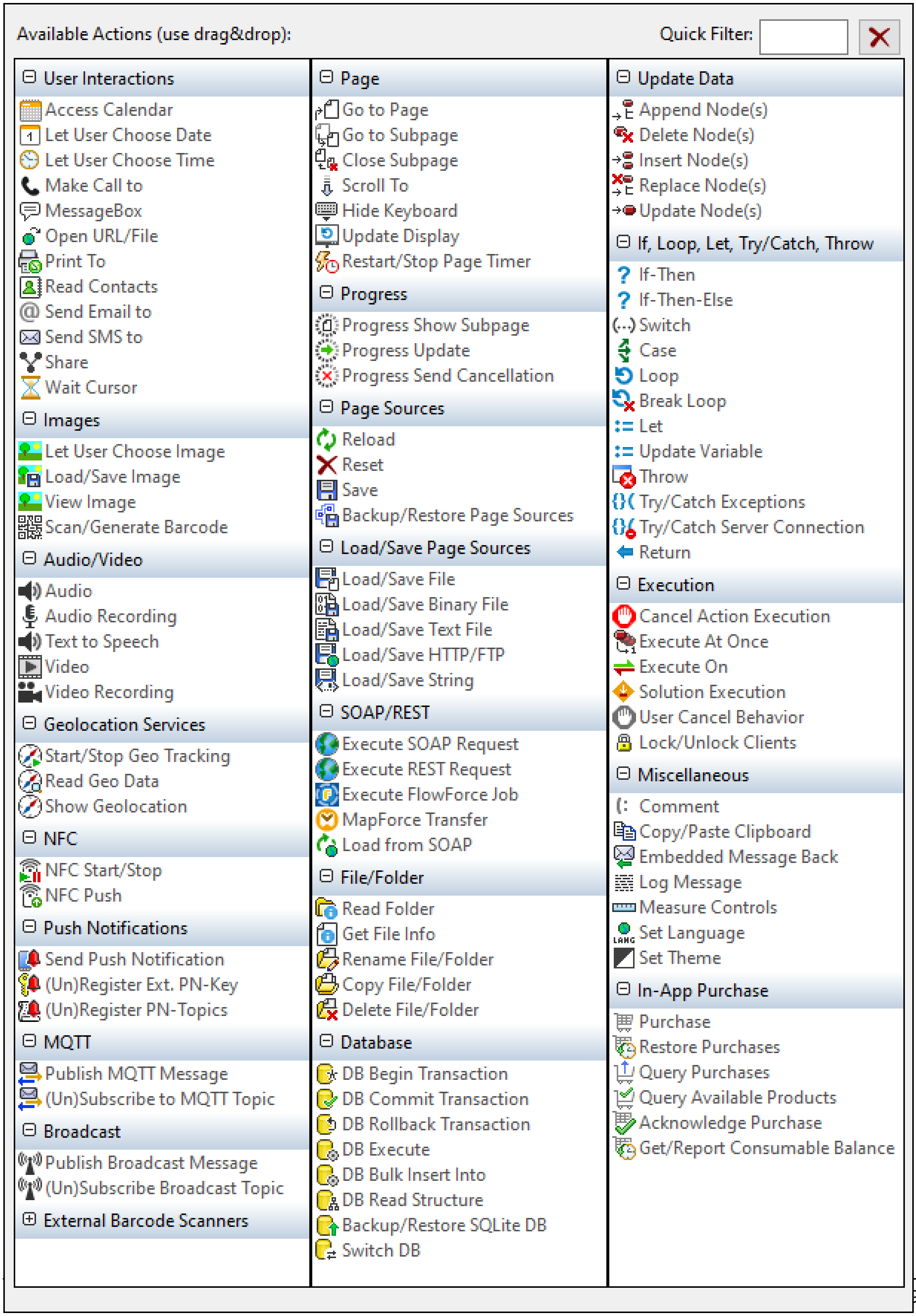
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box [679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions [1262].

## 10.15 File/Folder

The following actions are available in the File/Folder group of the Actions dialog (*screenshot below*):

- Read Folder 847
- Get File Info 849
- Rename File/Folder 850
- Copy File/Folder 851
- Delete File/Folder



Available Actions (use drag&drop):

Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

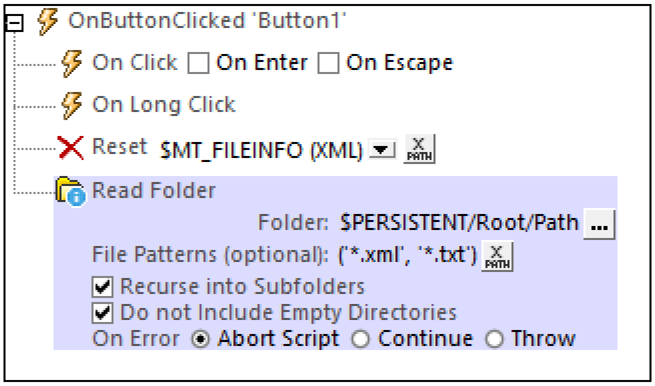
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events* 389 *and* *Control Events* 665.

## 10.15.1 Read Folder

When a Read Folder action (*see screenshot below*) is added to the design, the `$MT_FILEINFO` page source, which is structured as a set of repeating `File` elements, is created in the design. In the Read Folder action, you specify the folder that you want to read (*see screenshot and example below*). At run time, the contents of the specified folder (files and sub-folders) are read, and metadata information of each file and subfolder of the target folder is placed in a corresponding `File` element of the `$MT_FILEINFO` page source.


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Reset $MT_FILEINFO (XML)
Read Folder
Folder: $PERSISTENT/Root/Path
File Patterns (optional): ('*.xml', '*.txt')
☑ Recurse into Subfolders
☑ Do not Include Empty Directories
On Error ● Abort Script ○ Continue ○ Throw

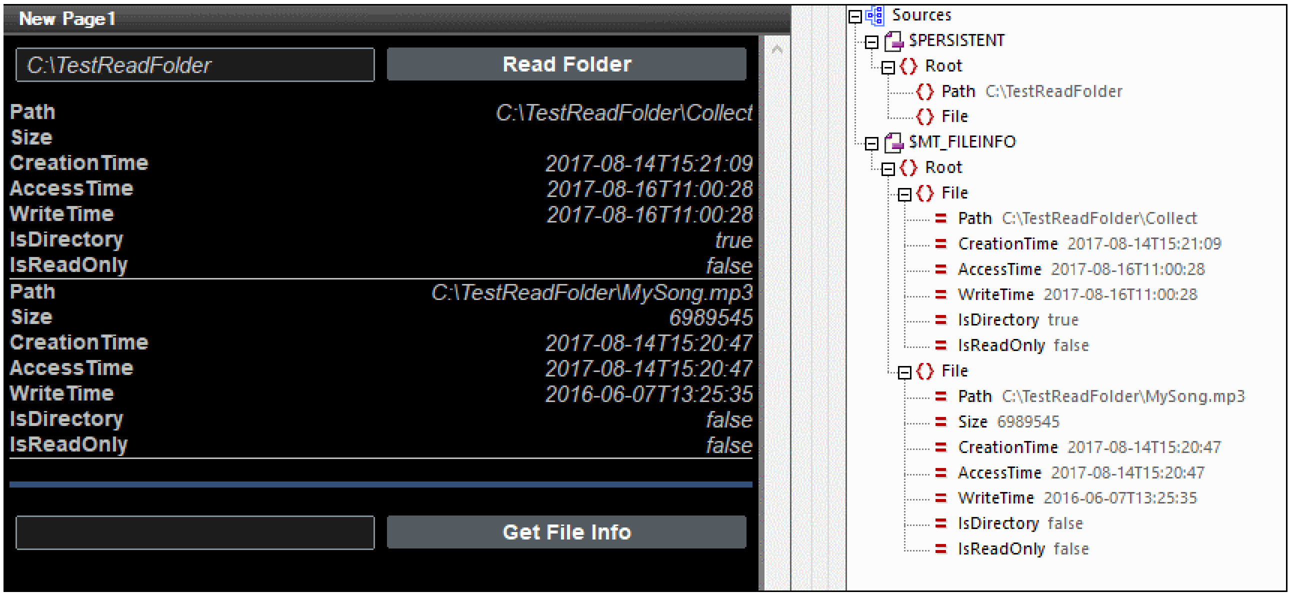
Note the following points:

- The folder can be specified either by browsing for it or by entering an XPath expression that evaluates to a string that is the path to the target folder.
- If the option to *Recurse into subfolders* is checked, then files and subfolders in recursive subfolders are read. Otherwise, only items in the specified folder are read.
- If the *Recurse into subfolders* option is checked, then the option to include empty directories becomes available. This enables information about empty folders to be included in the data that is read out.
- The File Pattern setting uses wildcards to filter which files of a folder are read. For example: `'*.mp3'` reads the details of all **`.mp3`** files in the folder; `'My*.*'` reads the details of all files that have names that begin with the characters **`My`** and take any suffix; `'*'` or `''` reads the entire folder. If recursion into subfolders has been enabled, then the pattern is applied recursively to subfolders.
- To enter multiple file patterns, the XPath expression must be a sequence of string items, each of which specifies a single pattern. For example: `("*.xml", "*.txt")`.
- While the Read Folder action provides information about files and subfolders of the target folder, it does not provide information about the target folder itself. For details of the folder itself, submit the name of the folder in the Get File Info 849 action.
- Each page contains one `$MT_FILEINFO` page source. So, if there are multiple Read Folder actions on a page, then, at run time, `$MT_FILEINFO` at any given time will contain information about the folder that was read by the last Read Action to have been triggered.
- The one `$MT_FILEINFO` page source is also populated with data obtained by the Get File Info 849 action. While the data read by the Read Folder action is passed to the **`File`** child elements of `$MT_FILEINFO/Root`, data read by the the Get File Info 849 action is passed to the attributes of `$MT_FILEINFO/Root`.

- The $MT_FILEINFO page source is created either when a Read Folder action or a Get File Info 849 action is added, whichever is added first.

## Example

An example file named **ReadFolderGetFileInfo.mtd** shows how to use the Read Folder action (*see screenshot of simulation below*). This file is available in the *(My) Documents* folder Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\Actions.



C:\TestReadFolder

| | |
|---|---|
| Path | C:\TestReadFolder\Collect |
| Size | |
| CreationTime | 2017-08-14T15:21:09 |
| AccessTime | 2017-08-16T11:00:28 |
| WriteTime | 2017-08-16T11:00:28 |
| IsDirectory | true |
| IsReadOnly | false |
| Path | C:\TestReadFolder\MySong.mp3 |
| Size | 6989545 |
| CreationTime | 2017-08-14T15:20:47 |
| AccessTime | 2017-08-14T15:20:47 |
| WriteTime | 2016-06-07T13:25:35 |
| IsDirectory | false |
| IsReadOnly | false |

```
Sources
  $PERSISTENT
    Root
      Path  C:\TestReadFolder
      File
  $MT_FILEINFO
    Root
      File
        Path  C:\TestReadFolder\Collect
        CreationTime  2017-08-14T15:21:09
        AccessTime  2017-08-16T11:00:28
        WriteTime  2017-08-16T11:00:28
        IsDirectory  true
        IsReadOnly  false
      File
        Path  C:\TestReadFolder\MySong.mp3
        Size  6989545
        CreationTime  2017-08-14T15:20:47
        AccessTime  2017-08-14T15:20:47
        WriteTime  2016-06-07T13:25:35
        IsDirectory  false
        IsReadOnly  false
```

The example works as follows:

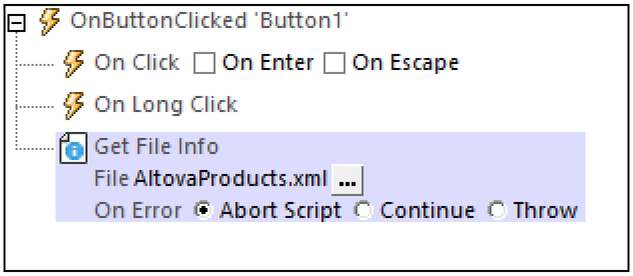
- The end user enters the name of a folder in the edit field to the left of the **Read Folder** button.
- The folder name is written to the **Path** node of the **$PERSISTENT** tree (*see screenshot*).
- The **Read Folder** button has a Read Folder action set for its OnButtonClick event. The action targets the folder stored in **$PERSISTENT/Root/Path**.
- On clicking **Read Folder**, data about the folder's items is read and passed to the **File** elements of $MT_FILEINFO.
- Each **File** element is displayed in the design as the repeating row of a table.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.15.2 Get File Info

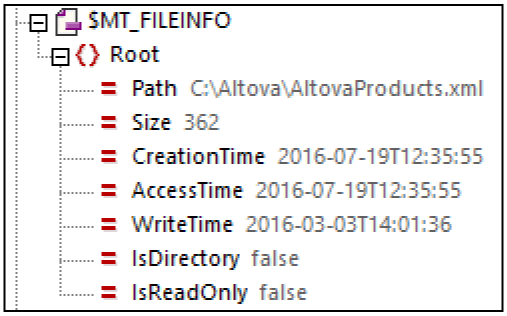
The Get File Info action (*screenshot below*) adds information about the specified file (or folder) to a page source called `$MT_FILEINFO`. An `$MT_FILEINFO` page source is created either when a Read Folder[847] action or a Get File Info action is added to the design, whichever is added first. In the settings of the action, select the file about which you want to get info.


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Get File Info
File AltovaProducts.xml ...
On Error ● Abort Script ○ Continue ○ Throw

Note that there is only one `$MT_FILEINFO` page source per page. So, if multiple Get File Info actions are triggered during the processing of a page at run time, then `$MT_FILEINFO` at any given time will contain information from the last Get File Info action that was triggered.

### Structure of the $MT_FILEINFO tree

The structure the `$MT_FILEINFO` tree is as shown in the screenshot below. The **`Root`** element has a number of **attributes** that will be filled with the file information of the file specified in the triggered action. Descriptions of the attributes are given below.


$MT_FILEINFO
Root
Path C:\Altova\AltovaProducts.xml
Size 362
CreationTime 2016-07-19T12:35:55
AccessTime 2016-07-19T12:35:55
WriteTime 2016-03-03T14:01:36
IsDirectory false
IsReadOnly false

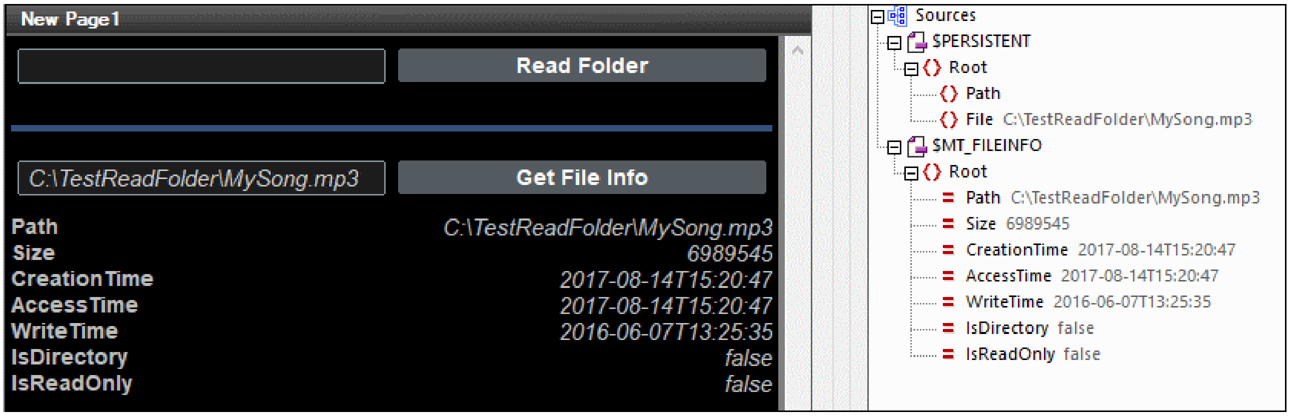
- *Path:* The full path of the file being reported.
- *Size:* The file size In bytes.
- *CreationTime:* The time when the file was created at its current location. If a file is copied to a new location, then the time at which it was copied will be the creation time. In such cases, the creation time could be later than the WriteTime.
- *AccessTime:* The time when the file was last accessed.
- *WriteTime:* The time when the file was last written to.
- *IsDirectory:* Can take a value of `true` or `false`.
- *IsReadOnly:* Can take a value of `true` or `false`.

**Note:** The data read by the Get File Info action is passed to the **attributes** of `$MT_FILEINFO/Root` (as described above). The `$MT_FILEINFO/Root` node, however, also has **child elements**: repeating **`File`**

elements, which receive data via another action, Read Folder 847. The Read Folder 847 action fills the **`File`** elements with metadata about files in a specified folder.

## Example

An example file named **`ReadFolderGetFileInfo.mtd`** shows how to use the Get Info action (*see screenshot of simulation below*). This file is available in the *(My) Documents* folder `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\Actions`.



C:\TestReadFolder\MySong.mp3

| | |
|---|---|
| Path | C:\TestReadFolder\MySong.mp3 |
| Size | 6989545 |
| CreationTime | 2017-08-14T15:20:47 |
| AccessTime | 2017-08-14T15:20:47 |
| WriteTime | 2016-06-07T13:25:35 |
| IsDirectory | false |
| IsReadOnly | false |

```
Sources
  $PERSISTENT
    Root
      Path
      File  C:\TestReadFolder\MySong.mp3
  $MT_FILEINFO
    Root
      Path  C:\TestReadFolder\MySong.mp3
      Size  6989545
      CreationTime  2017-08-14T15:20:47
      AccessTime  2017-08-14T15:20:47
      WriteTime  2016-06-07T13:25:35
      IsDirectory  false
      IsReadOnly  false
```

The example works as follows:

- The end user enters the name of a file in the edit field to the left of the **Get File Info** button.
- The file name is written to the **`Root/File`** node of the **`$PERSISTENT`** tree (*see screenshot*).
- The **Get File Info** button has a Get File Info action set for its `OnButtonClick` event that targets the file (or folder) stored in **`$PERSISTENT/Root/File`**.
- On clicking **Get File Info**, the information of the targeted file is read and passed to the `$MT_FILEINFO` page source as the values of the attributes of `$MT_FILEINFO/Root`.
- These attribute values are displayed in the cells of a static table.

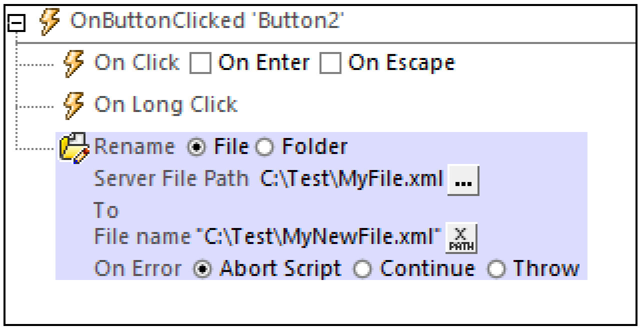
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-cache-update-dateTime()`**
**`mt-extract-file-extension()`**
**`mt-extract-file-name()`**

# 10.15.3 Rename File/Folder

Renames the selected file/folder on the client/server when the action is executed. Only one file/folder can be renamed per action. If you wish to rename multiple files/folders, use the action as many times as required.



OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

On Long Click

Rename ● File ○ Folder

Server File Path C:\Test\MyFile.xml ...

To

File name "C:\Test\MyNewFile.xml"

On Error ● Abort Script ○ Continue ○ Throw

- *What to rename:* Select whether you wish to rename a file or a folder and enter the path to the file or folder that you want to rename. In the Rename File dialog that appears, you can enter the path either directly or as an XPath expression that evaluates to the required path. (If you indicate, in the Rename File dialog, that the file resides on the server, then the field name will change from *File Path* to *Server File Path*.)
- *Renamed file/folder path:* Enter the path to the renamed file or folder. Note that: (i) a renamed folder can only be created if the parent folder exists, and (ii) if a file with the same name already exists at the specified location, then the new file silently replaces the original file.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

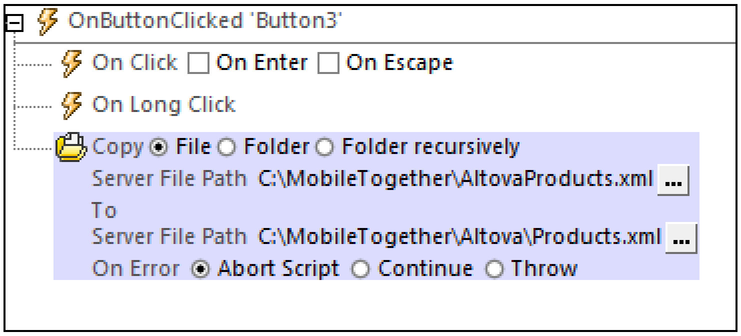
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable [907]. The Catch part of the Try/Catch action [907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action [907] for details.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions [1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box [679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions [1262].

# 10.15.4 Copy File/Folder

Copies the selected file/folder on the client/server when the action is executed. Only one file/folder can be copied per action. If you wish to copy descendant folders of the selected folder and their respective contents, select *Copy Folder Recursively*. If you wish to copy multiple files/folders, use the action as many times as required.



OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
Copy ● File ○ Folder ○ Folder recursively
Server File Path C:\MobileTogether\AltovaProducts.xml ...
To
Server File Path C:\MobileTogether\Altova\Products.xml ...
On Error ● Abort Script ○ Continue ○ Throw

- *What to copy:* Select the file or folder you want to copy. If you select a folder, only the files in the folder will be copied to the destination folder: the folder itself will not be copied, nor will any child folder be copied. To copy descendant folders of the selected folder and the contents of the descendant folders, select *Copy Folder Recursively*. In the Copy File dialog that appears, you can enter the path either directly or as an XPath expression that evaluates to the required path. (If you indicate, in the Copy File dialog, that the file resides on the server, then the field name will change from *File Path* to *Server File Path*.)
- *Destination location:* Enter the path of the location to which you want to copy the file or folder. If you are copying a file, the path must include the filename. If a file with the same name already exists at the specified location, then the new file silently replaces the original file.

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

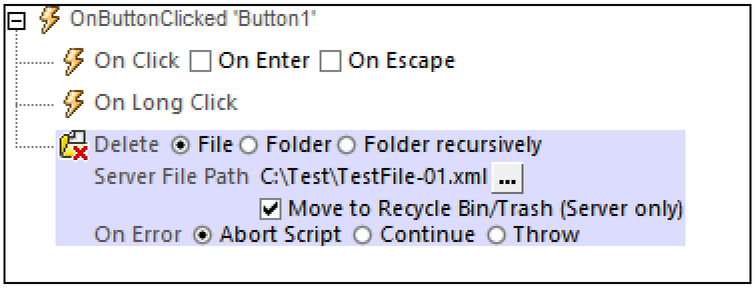
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.15.5 Delete File/Folder

Deletes the selected file/folder on the client/server when the action is executed. Only one file/folder can be deleted per action. If you wish to delete multiple files/folders, use the action repeatedly.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Delete ● File ○ Folder ○ Folder recursively
Server File Path C:\Test\TestFile-01.xml ...
☑ Move to Recycle Bin/Trash (Server only)
On Error ● Abort Script ○ Continue ○ Throw

- *What to delete:* Select whether you wish to delete: (i) a file, (ii) a folder (which is empty; if not empty, the folder will not be deleted ), or (iii) a folder recursively (the entire contents, including sub-folders, will be deleted).
- *File/Folder path:* The path to the file or folder to be deleted.
- *Move to Recycle Bin:* If this option is selected and a recycle bin exists on the server, then the file/folder is moved there. If the option is selected and no recycle bin exists on the server, then the file/folder is not deleted.

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

For additional information about file locations on clients and server, see Tree Data 354 and Load/Save Image 707.

### MobileTogether extension functions

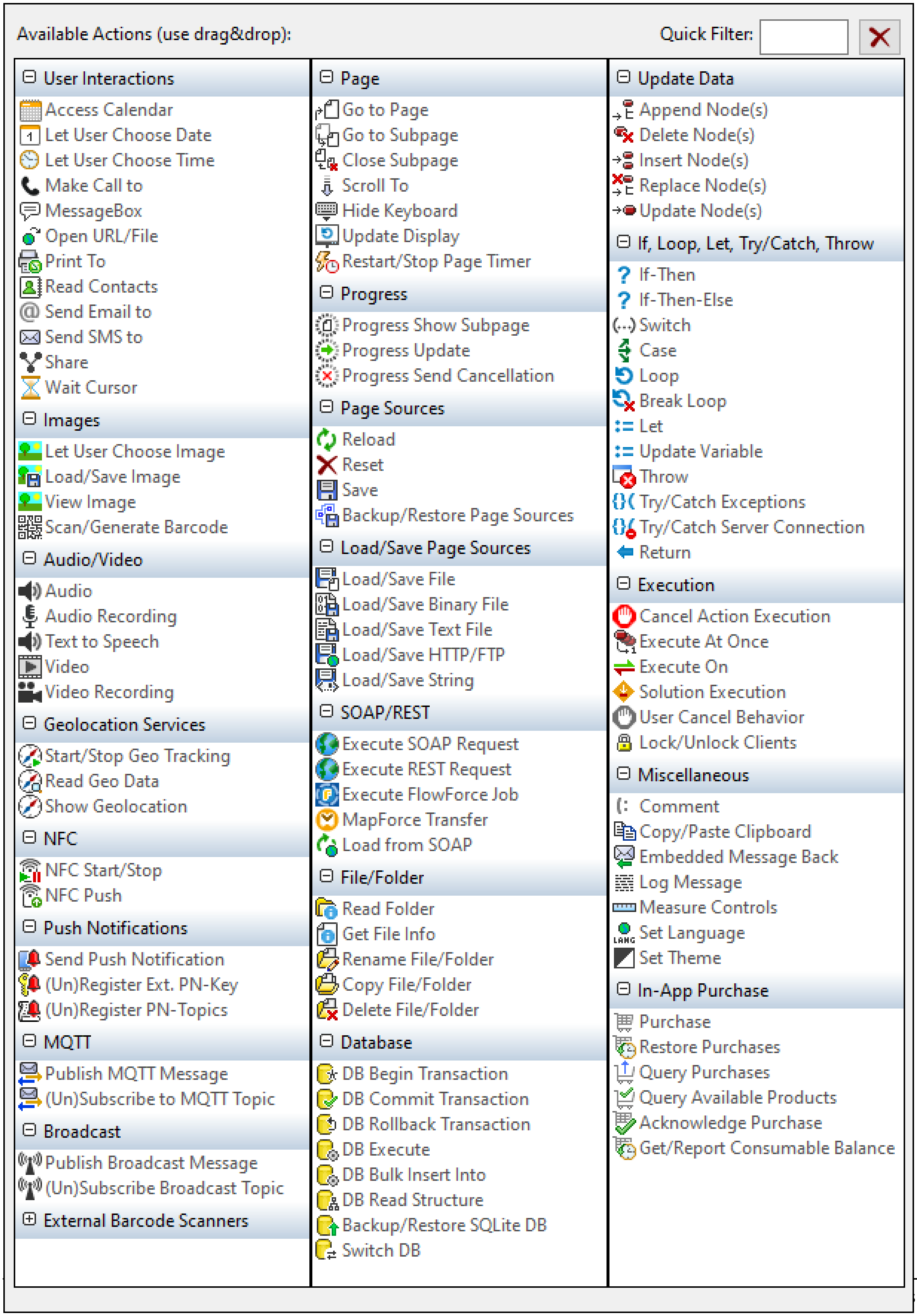
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

# 10.16 Database

The following actions are available in the Database group of the Actions dialog (*screenshot below*):

- DB Begin Transaction 856
- DB Execute 861
- DB Bulk Insert Into 865
- DB Commit Transaction 858
- DB Rollback Transaction 859
- DB Read Structure 867
- Backup/Restore SQLite DB 869
- Switch DB 871

**Note:** These actions are used to *interact* with data in DB page sources. They are not suitable for displaying data. If you wish to display data from a DB page source, then insert (into the design) a control 404 that is linked to a page source node 315. For information, see the sections about controls 404 and page sources 315. The tutorials 72 give you hands-on instructions about how to display page source data.



Available Actions (use drag&drop):

Quick Filter:

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

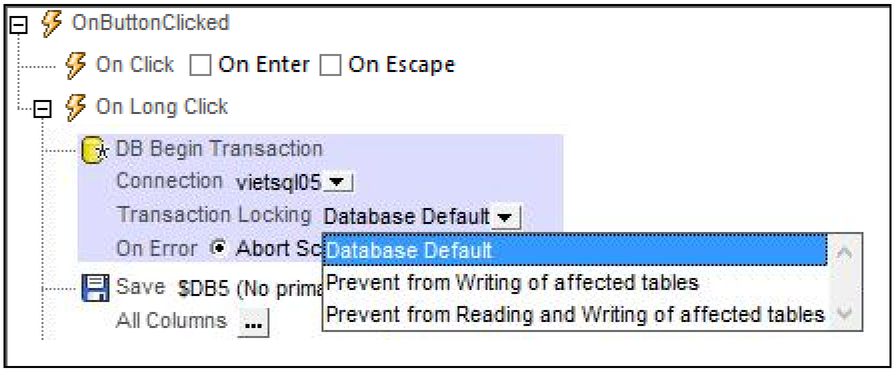
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events* 389 *and Control Events* 665.

## 10.16.1 DB Begin Transaction

When the event is triggered the DB Begin Transaction action begins a transaction with the data source selected in the *Connection* combo box. This combo box lists all the data sources of the project, and also offers the option of setting up an additional database connection specifically for use with the DB Begin Transaction action.



OnButtonClicked
On Click □ On Enter □ On Escape
On Long Click
DB Begin Transaction
Connection vietsql05
Transaction Locking Database Default
On Error ● Abort Sc
Save $DB5 (No prima
All Columns ...

Database Default
Prevent from Writing of affected tables
Prevent from Reading and Writing of affected tables

The *Transaction Locking* option specifies the level of protection, and enables you to ensure that data is not corrupted during write actions. The following options are available:

- *Database Default:* Collects DB-related default settings on the DB, server, and client.
- *Prevent from writing of affected tables:* The DB will not be written to if it is currently being written to via another connection. The Write-transaction will be deferred till the other Write-transaction has been completed; otherwise an error message will be displayed.
- *Prevent from reading and writing of affected tables:* The DB will not be read from or written to if it is currently being written to via another connection. The transaction will be deferred till the other transaction has been completed, or an error message will be displayed.
- *Exclusive:* This is an SQLite feature. When the `EXCLUSIVE` transaction is active, other connections are not able to read or write to the DB and get an error to the effect that the DB is locked.

If you connect to an SQLite database, a *Timeout (in seconds)* property becomes available. This enables you to specify a wait period for applying a write-lock. If no timeout period is specified, SQLite will abort immediately if a write-lock is not applied on transaction begin.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.

- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

**About DB Transactions**
For each DB access that needs a transaction, one is automatically created and closed afterwards. This might not be desirable for some setups. For example, when you have two DB page sources that you want to update atomically together: If both tables are saved successfully, then the transaction is committed, but rolled back otherwise. To accommodate this kind of situation, transactions can be created on a connection basis.

If you begin a transaction 856, all DB operations belonging to the same DB connection will use this transaction.

Committing a transaction 858 makes changes visible to the world outside your transaction. Changes can be rolled back 859. In this case, even if you have done a Save on your page source, the changes won't be visible after a rollback! Note that any transaction that is not closed (committed or rolled back) when the end of the action tree has been reached will be rolled back automatically! A warning to this effect will be displayed in the Messages window.

It is important to bear in mind that, while the behavior above refers to explicit transaction actions, this behavior also applies to all DB operations that use the same connection as the transaction.

## MobileTogether extension functions

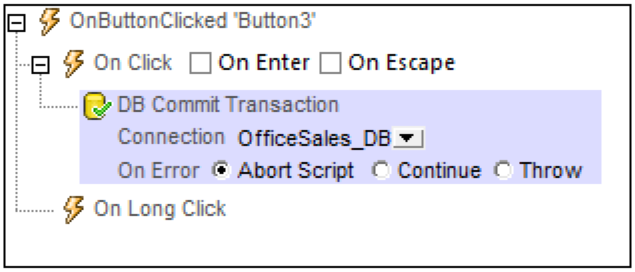
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

```
mt-available-db-connection-names()
mt-db-any-changed-fields()
mt-db-any-changed-rows()
mt-db-deleted-original-fields()
mt-db-deleted-original-rows()
mt-db-file-path()
mt-db-modified-fields()
mt-db-modified-rows()
mt-db-new-fields()
mt-db-new-rows()
mt-db-original row()
mt-external-error-code()
```

**mt-external-error-text()**

## 10.16.2 DB Commit Transaction

When the event is triggered the DB Commit Transaction action commits a transaction to the page source selected in the *Connection* combo box. This combo box lists all the page sources of the project, and also offers the option of setting up an additional database connection specifically for use with the DB Commit Transaction action.



OnButtonClicked 'Button3'

On Click □ On Enter □ On Escape

DB Commit Transaction

Connection OfficeSales_DB

On Error ● Abort Script ○ Continue ○ Throw

On Long Click

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

**About DB Transactions**

For each DB access that needs a transaction, one is automatically created and closed afterwards. This might not be desirable for some setups. For example, when you have two DB page sources that you want to update atomically together: If both tables are saved successfully, then the transaction is committed, but rolled back otherwise. To accommodate this kind of situation, transactions can be created on a connection basis.

If you begin a transaction 856, all DB operations belonging to the same DB connection will use this transaction.

Committing a transaction 858 makes changes visible to the world outside your transaction. Changes can be rolled back 859. In this case, even if you have done a Save on your page source, the changes won't be visible after a rollback! Note that any transaction that is not closed (committed or rolled back) when the end of the action tree has been reached will be rolled back automatically! A warning to this effect will be displayed in the Messages window.

It is important to bear in mind that, while the behavior above refers to explicit transaction actions, this behavior also applies to all DB operations that use the same connection as the transaction.

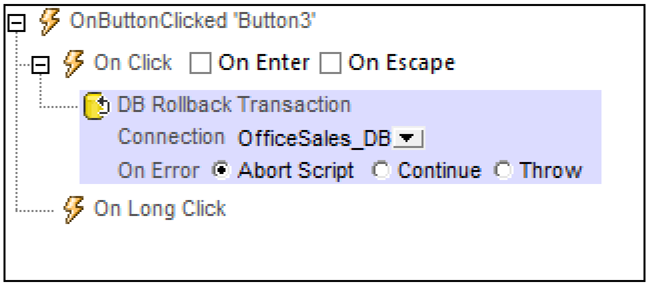
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

```
mt-available-db-connection-names()
mt-db-any-changed-fields()
mt-db-any-changed-rows()
mt-db-deleted-original-fields()
mt-db-deleted-original-rows()
mt-db-file-path()
mt-db-modified-fields()
mt-db-modified-rows()
mt-db-new-fields()
mt-db-new-rows()
mt-db-original row()
mt-external-error-code()
mt-external-error-text()
```

## 10.16.3 DB Rollback Transaction

When the event is triggered the DB Rollback Transaction action rolls back a transaction on the page source selected in the *Connection* combo box. This combo box lists all the page sources of the project, and also offers the option of setting up an additional database connection.



OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
DB Rollback Transaction
Connection OfficeSales_DB
On Error ● Abort Script ○ Continue ○ Throw
On Long Click

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—

and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

**About DB Transactions**
For each DB access that needs a transaction, one is automatically created and closed afterwards. This might not be desirable for some setups. For example, when you have two DB page sources that you want to update atomically together: If both tables are saved successfully, then the transaction is committed, but rolled back otherwise. To accommodate this kind of situation, transactions can be created on a connection basis.

If you begin a transaction 856, all DB operations belonging to the same DB connection will use this transaction.

Committing a transaction 858 makes changes visible to the world outside your transaction. Changes can be rolled back 859. In this case, even if you have done a Save on your page source, the changes won't be visible after a rollback! Note that any transaction that is not closed (committed or rolled back) when the end of the action tree has been reached will be rolled back automatically! A warning to this effect will be displayed in the Messages window.

It is important to bear in mind that, while the behavior above refers to explicit transaction actions, this behavior also applies to all DB operations that use the same connection as the transaction.

## MobileTogether extension functions

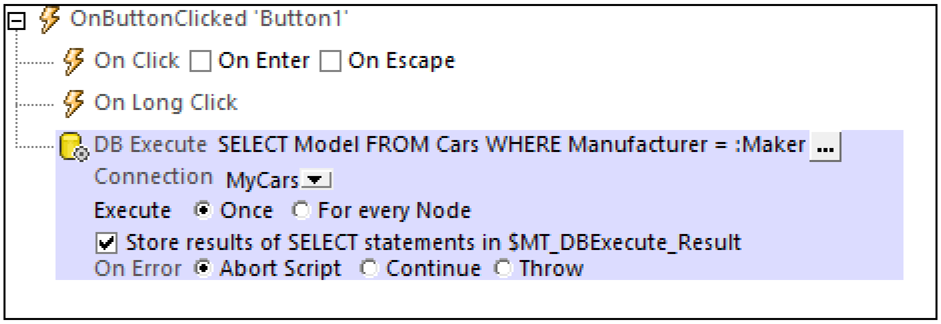
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-available-db-connection-names()`**
**`mt-db-any-changed-fields()`**
**`mt-db-any-changed-rows()`**
**`mt-db-deleted-original-fields()`**
**`mt-db-deleted-original-rows()`**
**`mt-db-file-path()`**
**`mt-db-modified-fields()`**
**`mt-db-modified-rows()`**

```
mt-db-new-fields()
mt-db-new-rows()
mt-db-original row()
mt-external-error-code()
mt-external-error-text()
```

## 10.16.4 DB Execute

When the event is triggered, the DB Execute action executes the action's SQL statement on the data source selected in the *Connection* combo box. This combo box lists all the data sources of the project, and also offers the option of setting up an additional database connection specifically for use with the DB Execute action. If the *Store results in* `$MT_DBExecute_Result` check box is selected, then the results are stored in the `$MT_DBExecute_Result` 1300 variable. This variable can then be used in XPath expressions elsewhere on the page to provide the result of the DB Execute action.



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

DB Execute SELECT Model FROM Cars WHERE Manufacturer = :Maker ...

Connection MyCars

Execute ● Once ○ For every Node

☑ Store results of SELECT statements in $MT_DBExecute_Result

On Error ● Abort Script ○ Continue ○ Throw

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

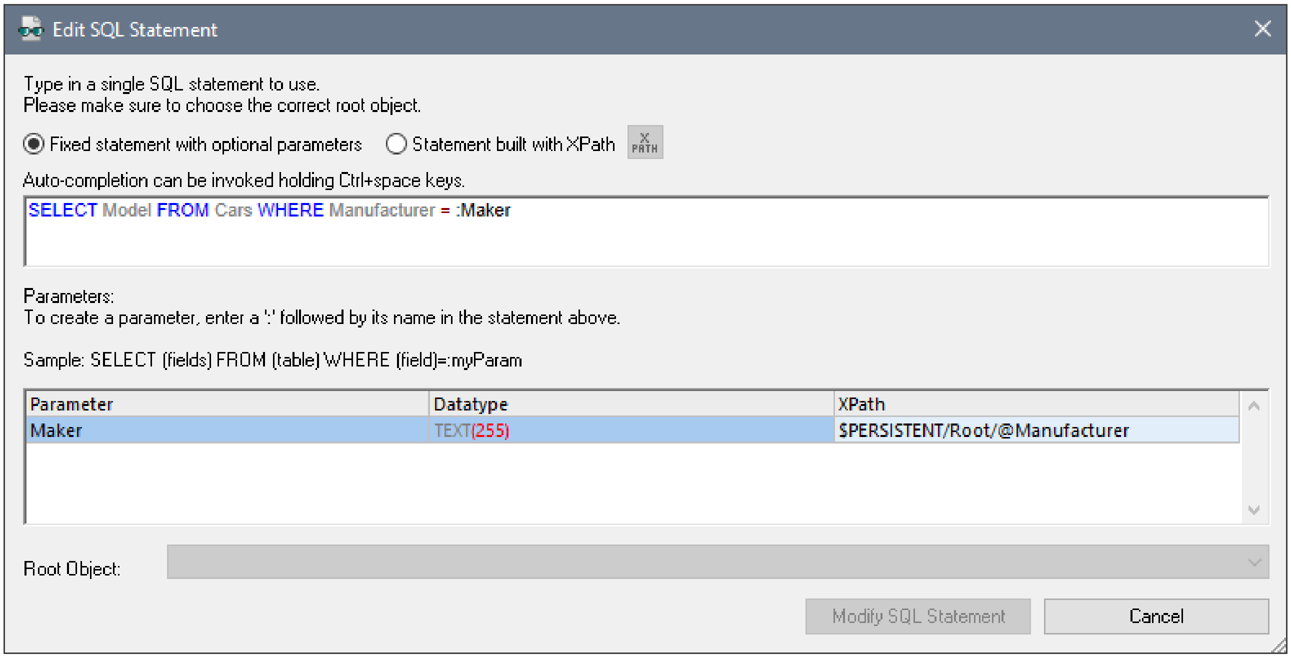
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

For more information about using the action, see the section Page Design | Database | The DB Execute Action 1040.

**Note:** The DB Execute action is used to *interact* with data in DB page sources. It is not the ideal mechanism for displaying data. If you wish to display data from a DB page source node, then insert (into the design) a control[404] that is linked to a page source node[315]. For information, see the sections about controls[404] and page sources[315]. The tutorials[72] give you hands-on instructions about how to display page source data.

## The SQL statement

To enter or edit the SQL statement, click the **Additional Dialog** button. This displays the Edit SQL Statement dialog (*screenshot below*). The Root Object at the bottom of the dialog is selected automatically and is based on the selection in the Connection combo box. The *Root Object* field cannot be edited. Before proceeding, make sure that this is the root object you want.



Type in a single SQL statement to use.
Please make sure to choose the correct root object.

● Fixed statement with optional parameters ○ Statement built with XPath

Auto-completion can be invoked holding Ctrl+space keys.

```
SELECT Model FROM Cars WHERE Manufacturer = :Maker
```

Parameters:
To create a parameter, enter a ':' followed by its name in the statement above.

Sample: SELECT (fields) FROM (table) WHERE (field)=:myParam

| Parameter | Datatype | XPath |
|---|---|---|
| Maker | TEXT(255) | $PERSISTENT/Root/@Manufacturer |

Root Object:

*Fixed statement with optional parameters*

To enter an SQL statement, select *Fixed statement with optional parameters*, and enter the SQL statement. The use of parameters in the SQL statement provides more flexibility. For example, in the screenshot above, instead of entering a fixed value for the `WHERE` clause, a parameter name `Maker` is used to supply the value of a node in an XML page source. In the first line below, a fixed value is used; in the second line, the parameter `Maker` is used.

```
WHERE Manufacturer= 'BMW'
WHERE Manufacturer= :Maker
```

To use a parameter, write the parameter name prefixed by a colon (`:`)in the SQL statement where you want to use it. As soon as you enter the first character after the colon, an entry is created for the parameter in the Parameters pane. Next, in the Parameters pane, enter an XPath expression to provide the value of the parameter. You can enter as many parameters as you like.

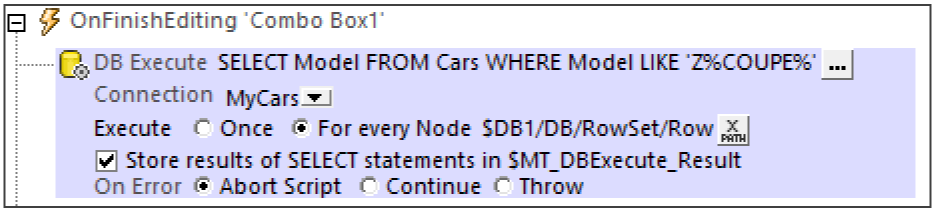
**Note:** In the SQL statement, column and table names from the source database are used, since the SQL statement directly queries the DB. In the XPath expression of parameters, however, you must use the names of nodes in page source trees (`Row`, `RowSet`, etc), since it is these trees in which values related to the design are stored.

*Statement built with XPath*
You can also use XPath to build an SQL statement. Select *Statement built with XPath*, and enter the XPath expression that generates the required SQL statement. The advantage of this is that it provides greater flexibility in creating the SQL statement. For example, you can include design tree nodes, other XPath constructs, and end user input to calculate and generate parts of the SQL statement.

To build an SQL statement using XPath, select *Statement built with XPath*. In the Edit XPath/XQuery Expression dialog[1244] that appears, enter the XPath expression and click **OK**.

## Execute once or for every node

The SQL statement can be executed once on the data source, or it can be executed on all the nodes of a custom defined nodeset. If you select the latter option, you need to enter an XPath expression that generates the nodeset. The SQL statement will then be executed for each node in this nodeset. Additionally, you can query the value of the current node of the nodeset by using the variable `$MT_TargetNode`[1304]. This variable can be used, for example, in the definition of a parameter used in the SQL statement (*see "SQL statements with parameters" above*).

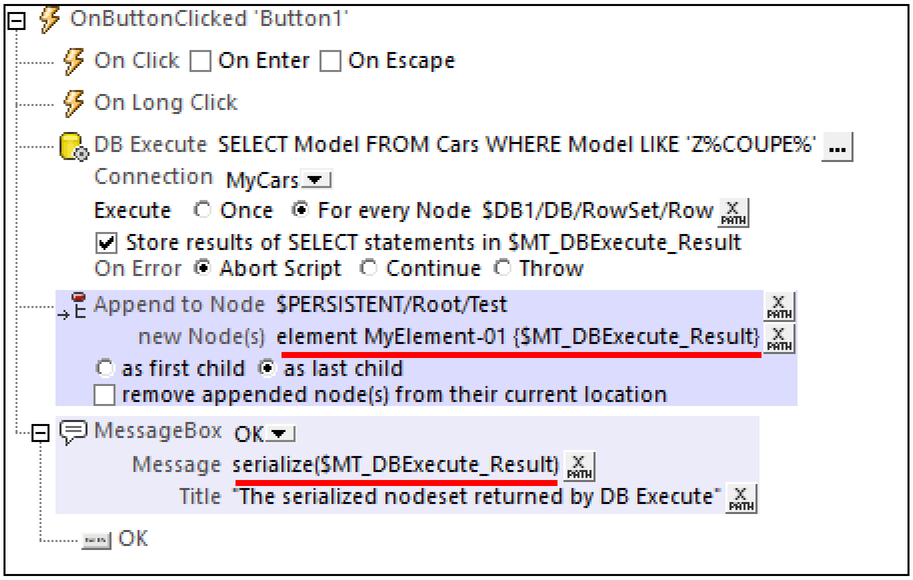


OnFinishEditing 'Combo Box1'
DB Execute SELECT Model FROM Cars WHERE Model LIKE 'Z%COUPE%'
Connection MyCars
Execute ○ Once ● For every Node $DB1/DB/RowSet/Row
☑ Store results of SELECT statements in $MT_DBExecute_Result
On Error ● Abort Script ○ Continue ○ Throw

## The **$MT_DBExecute_Result** variable

The nodeset or other value returned by (the SQL statement of) the DB Execute action is stored in MobileTogether Designer's built-in variable **`$MT_DBExecute_Result`**. This variable stores the result of the last DB Execute action of the project, and can be used in XPath expressions in other locations in the project.

If the DB Execute action returns a nodeset, you can construct an element and insert the nodeset as a child of the constructed element. Alternatively, you can serialize the nodeset by using the **`serialize()`** function, like this: **`serialize($MT_DBExecute_Result)`**. The two XPath expressions are shown underlined in red in the screenshots below.



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

DB Execute SELECT Model FROM Cars WHERE Model LIKE 'Z%COUPE%' ...

Connection MyCars

Execute ○ Once ● For every Node $DB1/DB/RowSet/Row

☑ Store results of SELECT statements in $MT_DBExecute_Result

On Error ● Abort Script ○ Continue ○ Throw

Append to Node $PERSISTENT/Root/Test

new Node(s) element MyElement-01 {$MT_DBExecute_Result}

○ as first child ● as last child

□ remove appended node(s) from their current location

MessageBox OK

Message serialize($MT_DBExecute_Result)

Title "The serialized nodeset returned by DB Execute"

OK

If you want to access a specific node in the nodeset, you can access it with an XPath expression. For example, consider the DB Execute action in the screenshot above. Let us say that the **SELECT** statement returns the following nodeset:

```
<DB>
   <RowSet>
       <Row Model="Z3 COUPE 2014"/>
       <Row Model="Z3 COUPE 2015"/>
       <Row Model="Z3 COUPE 2016"/>
       <Row Model="Z4 3.0 SI COUPE 2014"/>
       <Row Model="Z4 COUPE 2015"/>
   </RowSet>
</DB>
```

- To access the model name of the first car in the returned nodeset, the XPath expression would be: `$MT_DBExecute_Result/DB/RowSet/Row[1]/@Model`.
- To access the row of the car with the year 2016 in its model name, the XPath expression would be: `$MT_DBExecute_Result/DB/RowSet/Row[@Model[contains(. , '2016')]]`.

## MobileTogether extension functions

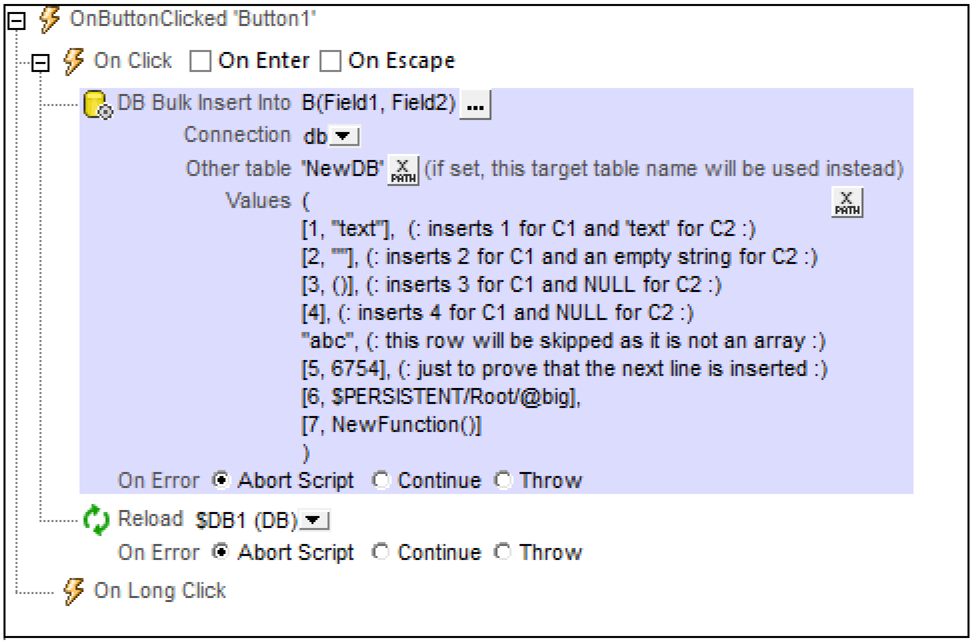
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**mt-available-db-connection-names()**
**mt-db-any-changed-fields()**
**mt-db-any-changed-rows()**
**mt-db-deleted-original-fields()**
**mt-db-deleted-original-rows()**

**`mt-db-file-path()`**
**`mt-db-modified-fields()`**
**`mt-db-modified-rows()`**
**`mt-db-new-fields()`**
**`mt-db-new-rows()`**
**`mt-db-original row()`**
**`mt-external-error-code()`**
**`mt-external-error-text()`**

## 10.16.5 DB Bulk Insert Into

The DB Bulk Insert Into action appends the data submitted via the XPath expression of the *Values* field as new rows into the DB table that is selected in the *DB Bulk Insert Into* setting (*see screenshot below*).



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
DB Bulk Insert Into B(Field1, Field2) ...
Connection db
Other table 'NewDB' (if set, this target table name will be used instead)
Values

```
(
[1, "text"],  (: inserts 1 for C1 and 'text' for C2 :)
[2, ""], (: inserts 2 for C1 and an empty string for C2 :)
[3, ()], (: inserts 3 for C1 and NULL for C2 :)
[4], (: inserts 4 for C1 and NULL for C2 :)
"abc", (: this row will be skipped as it is not an array :)
[5, 6754], (: just to prove that the next line is inserted :)
[6, $PERSISTENT/Root/@big],
[7, NewFunction()]
)
```

On Error ● Abort Script ○ Continue ○ Throw
Reload $DB1 (DB)
On Error ● Abort Script ○ Continue ○ Throw
On Long Click

- *DB Bulk Insert Into*: When selecting the DB table into which to insert, you specify the DB connection method 960 and then select the table into which the new rows are to be inserted. The new rows will be appended to the existing rows of the table. The selected table is listed in the *DB Bulk Insert Into* field, together with its columns. For example, in the screenshot above, the selected table is named `B`, and it has two fields, named `Field1` and `Field2`.
- *Other table*: A table other than the one selected in *DB Bulk Insert Into* can be specified via an XPath expression. The new rows will be inserted into this table as well. This table must already exist, and it must contain columns with the same names as the table columns selected in *DB Bulk Insert Into.* It can contain additional columns if these have default values or are nullable. Also, the datatype of each column must match the datatype of the corresponding column of the table selected in *DB Bulk Insert Into.* In the screenshot above, the new rows will be inserted into the table `NewDB`. If the insertion is to be successful, the `NewDB` table must have two columns with the names *Field1* and *Field2*, respectively. The datatypes too must match: the first column being of a number datatype, the second column being of a string datatype. If submitted values do not match a column's datatype, then a conversion is attempted.

- *Values*: The XPath expression of the *Values* field must return a sequence of arrays, where each array represents a row and where each value in an array represents a column value. In the screenshot above, each array is placed on a new line. Notice the various ways the array items are instantiated.

In the screenshot above, we have used a Reload action[801] to update the DB page source that contains the modified table.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable[907]. The Catch part of the Try/Catch action[907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action[907] for details.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

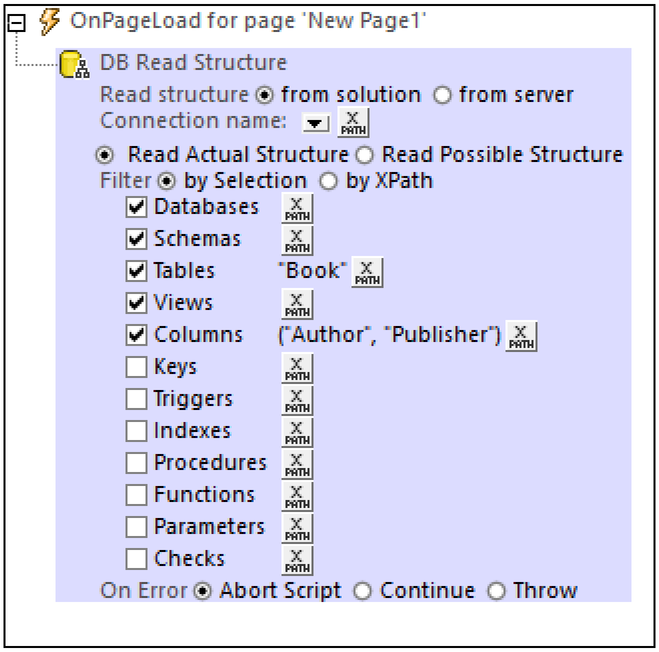
```
mt-available-db-connection-names()
mt-db-any-changed-fields()
mt-db-any-changed-rows()
mt-db-deleted-original-fields()
mt-db-deleted-original-rows()
mt-db-file-path()
mt-db-modified-fields()
mt-db-modified-rows()
mt-db-new-fields()
mt-db-new-rows()
mt-db-original row()
mt-external-error-code()
mt-external-error-text()
```

## 10.16.6 DB Read Structure

The DB Read Structure action enables data to be read from a specified database and stored in a page source named `$MT_DBSTRUCTURE`. This page source is filled exclusively by data that is obtained when the DB Read Structure action is executed.

### Defining the DB Read Structure action

When the DB Read Structure action is dropped in the Event pane, a `$MT_DBSTRUCTURE` page source is added to the design (visible in the Page Sources Pane[270]). The DB that will be read is defined in the action's settings (*screenshot below*). These settings are described below.



OnPageLoad for page 'New Page1'

DB Read Structure

Read structure ● from solution ○ from server

Connection name:

● Read Actual Structure ○ Read Possible Structure

Filter ● by Selection ○ by XPath

☑ Databases

☑ Schemas

☑ Tables "Book"

☑ Views

☑ Columns ("Author", "Publisher")

□ Keys

□ Triggers

□ Indexes

□ Procedures

□ Functions

□ Parameters

□ Checks

On Error ● Abort Script ○ Continue ○ Throw

*Read structure from*

Specifies the location of the DB structure; solution or server. The structure can be one of the DB page sources in the solution, or it can be a DB that is accessed via a connection stored on MobileTogether Server. For information about stored DB connections, see the description of server-side DB connections in the MobileTogether Server user manual.

**Note**: Server-side DB connections are available only on Windows-based MobileTogether Servers. As a result, on Linux-based MobileTogether Servers, you are restricted to reading DB structures that are contained in the solution.

*Connection name*

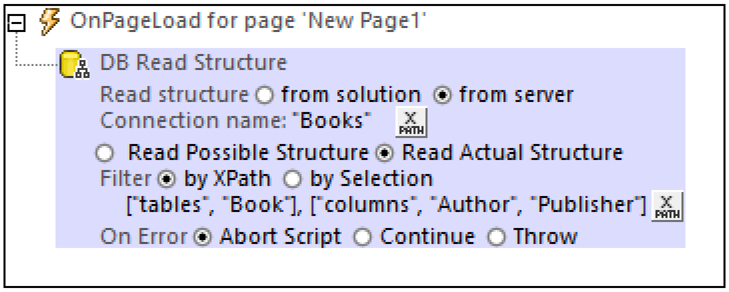
The connection name can be entered as an XPath string value *(see screenshot below)*. If the structure's location has been specified to be the solution (*see previous point*), then the connection name is also available for selection in a combo box.

*Read Actual/Possible Structure:*
The `$MT_DBSTRUCTURE` page source has a structure that is a superset containing components that are available across various types of DBs.

- *Read Actual Structure:* Reads the structure of the given DB connection. You can select components to read, as well as filter these components by name (*see screenshot above*).
- *Read Possible Structure:* That subset of nodes of the `$MT_DBSTRUCTURE` page source will be filled that matches the structure of the given DB connection and for which data can be returned. Tables of the DB structure read in this way are not identified by their names.

*Filter*
This option is displayed only if the *Read Actual Structure* option *(see previous option)* is selected. It enables you to filter what DB components to read. You can filter the components in one of the following ways:

- *By Selection:* Select the check boxes of the components to be read *(see screenshot above).* To further filter the selected component by name, specify an XPath expression which is a sequence of strings that gives the names of the component to be read. If a particular component has ancestor-type components, then the ancestor-type components will automatically also be read. For example, if a column has been selected, then the column's table ancestor will automatically also be read.
- *By XPath:* The XPath expression must be a sequence of arrays *(see screenshot below)*. The first item in each array is the component type to read (`tables`, `columns`, etc); these items are known as keywords, and available keywords are displayed in a popup that appears when you hover over the option's **`XPath`** button; keywords are case-insensitive. The subsequent items of the array (second item onwards) are the *names* of that component type to read. For example, in the XPath expression shown in the screenshot below, the columns named **`Author`** and **`Publisher`** of tables named **`Book`** are read.



OnPageLoad for page 'New Page1'
DB Read Structure
Read structure ○ from solution ● from server
Connection name: "Books"
○ Read Possible Structure ● Read Actual Structure
Filter ● by XPath ○ by Selection
["tables", "Book"], ["columns", "Author", "Publisher"]
On Error ● Abort Script ○ Continue ○ Throw

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for

details.

### What happens at runtime

At runtime, the specified DB is read, and the nodes of the `$MT_DBSTRUCTURE` page source are filled with data from the DB. The data in this page source can now be used in the design.

**Note:** A MobileTogether XPath extension function[1262] named `mt-available-db-connection-names` can be used to get the names of all available DB connections, either from the solution or the server.

### Simulations

If you use the server for simulation, make sure that the server settings in MobileTogether Designer are correctly set[1663] and that the DB is available in the server side solution's working directory. *See the section Simulation on Server[1362] for more information.*

If you run a simulation directly in MobileTogether Designer, the data that will be used in the simulation will come from the DB that is specified in the *Read DB-Structure Simulation* setting (which is available in the Simulation 2 tab of the Options dialog[1663]).

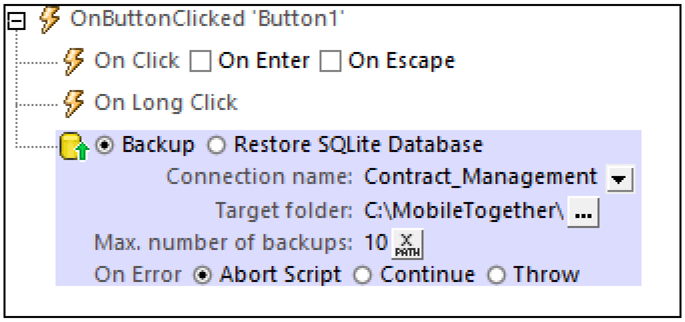
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-available-db-connection-names()`**
**`mt-db-any-changed-fields()`**
**`mt-db-any-changed-rows()`**
**`mt-db-deleted-original-fields()`**
**`mt-db-deleted-original-rows()`**
**`mt-db-file-path()`**
**`mt-db-modified-fields()`**
**`mt-db-modified-rows()`**
**`mt-db-new-fields()`**
**`mt-db-new-rows()`**
**`mt-db-original row()`**
**`mt-external-error-code()`**
**`mt-external-error-text()`**

## 10.16.7 Backup/Restore SQLite DB

This action *(screenshot below)* enables you to backup an SQLite database multiple times to a folder you designate. You can subsequently restore the SQLite database from one of these backups. This feature is available only on MobileTogether Server Advanced Edition.



OnButtonClicked 'Button1'
- On Click □ On Enter □ On Escape
- On Long Click
- ● Backup ○ Restore SQLite Database
  - Connection name: Contract_Management
  - Target folder: C:\MobileTogether\
  - Max. number of backups: 10
  - On Error ● Abort Script ○ Continue ○ Throw

To set backups, do the following:

1. Select *Backup.*
2. Select the SQLite DB connection that you want to back up.
3. Select the folder where the backups should be stored. This can be a path that is relative to the server-side solution's working directory or an absolute path. The filename of the backed up SQLite database will be generated automatically and is a concatenation of (i) the DB connection name that you entered in Step 2 (for example, *Contract_Management* in the screenshot above) and (ii) the current timestamp (in the format `YYYY-MM-DD HH-MM-SS`). So, for example, a possible filename would be: **`Contract_Management 2021-06-18_10-30-24`**, where the numbers are the date and time.
4. Optionally, enter the maximum number of backups. After this number is exceeded, the oldest backup will be deleted. If no value is set or if a value of `0` is set, then an unlimited number of backups is allowed.

To restore the SQLite database from a backup, do the following:

1. Select *Restore.*
2. Select the connection to the SQLite DB that you want to restore.
3. Select the relative or absolute path of the backup file from which to restore. Relative paths must be be relative to the server-side solution's working directory.

*Note:*

- After a backup has been created, the path to it can be obtained by calling the **`mt-last-file-path()`** 1262 function. Note that this function returns the full file path.
- When a SQLite database is restored via the Restore action, any new data in the replaced database file (since the last backup) will be lost. If you want to keep this data make sure that you back up the database before restoring it.
- After a backed up file is restored via the Restore action, the backup file is not deleted. After a Restore action, the **`mt-last-file-path()`** 1262 function returns the full path of the backup file (that was used for the restore).
- If the DB is locked when a restore is attempted, an error is returned. There are no retries or timeouts.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.

- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

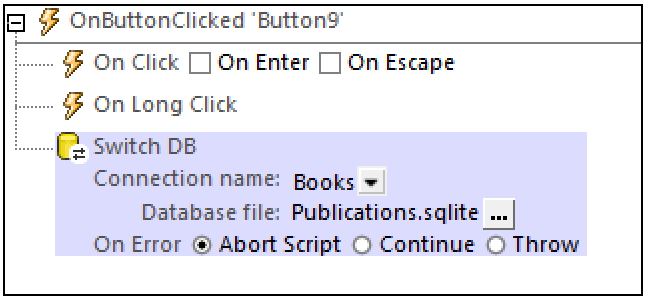
**`mt-available-db-connection-names()`**
**`mt-external-error-code()`**
**`mt-external-error-text()`**

## 10.16.8 Switch DB

The Switch DB action works for file-based database (SQLite or MS Access) and enables you to link a different database to a DB connection. In this way you can easily change the data of a DB page source. In the Switch DB action, you select the DB page source (data connection) for which you want to change the data, and then select the new data file. When the switch is made at run time, the affected page sources are reset. In MobileTogether Designer simulations, the reset page sources are listed in the Messages window.

Note the following points:

- The Switch DB feature works only with file-based DBs. Both the original and replacing databases must be either SQLite or MS Access, and they must match the data structure of the page source (data connection).
- The Switch DB action is compatible with **MobileTogether Server Advanced Edition only**, and not with the standard edition .
- In MobileTogether Designer, you cannot set a DB switch on a page source that has been set to *Auto-load on first use*. If you try to do this, then MobileTogether Designer will list the page source in the Messages window together with a link to it. You can click the link to quickly jump to the page source and change the *Auto-load* setting.
- If the client session times out or if the solution is suspended, the solution will restart with the switched data. To get back to the original data source in the course of the current workflow, you must add another DB Switch action that is set to the original database. A solution restart will, of course, start the solution with the original database.



OnButtonClicked 'Button9'

On Click □ On Enter □ On Escape

On Long Click

Switch DB

Connection name: Books

Database file: Publications.sqlite

On Error ● Abort Script ○ Continue ○ Throw

*Connection name*
The *Connection Name* setting is the name of a file-based DB page source. The combo box contains a list of connections to file-based DBs in the design.

*Database file*
The *Database File* setting is the path to the database that you want to switch to for the selected database connection. It should have a data structure that is the same as that of the selected page source.

*Error processing*
The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable 907. The Catch part of the Try/Catch action 907 is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action 907 for details.

## MobileTogether extension functions

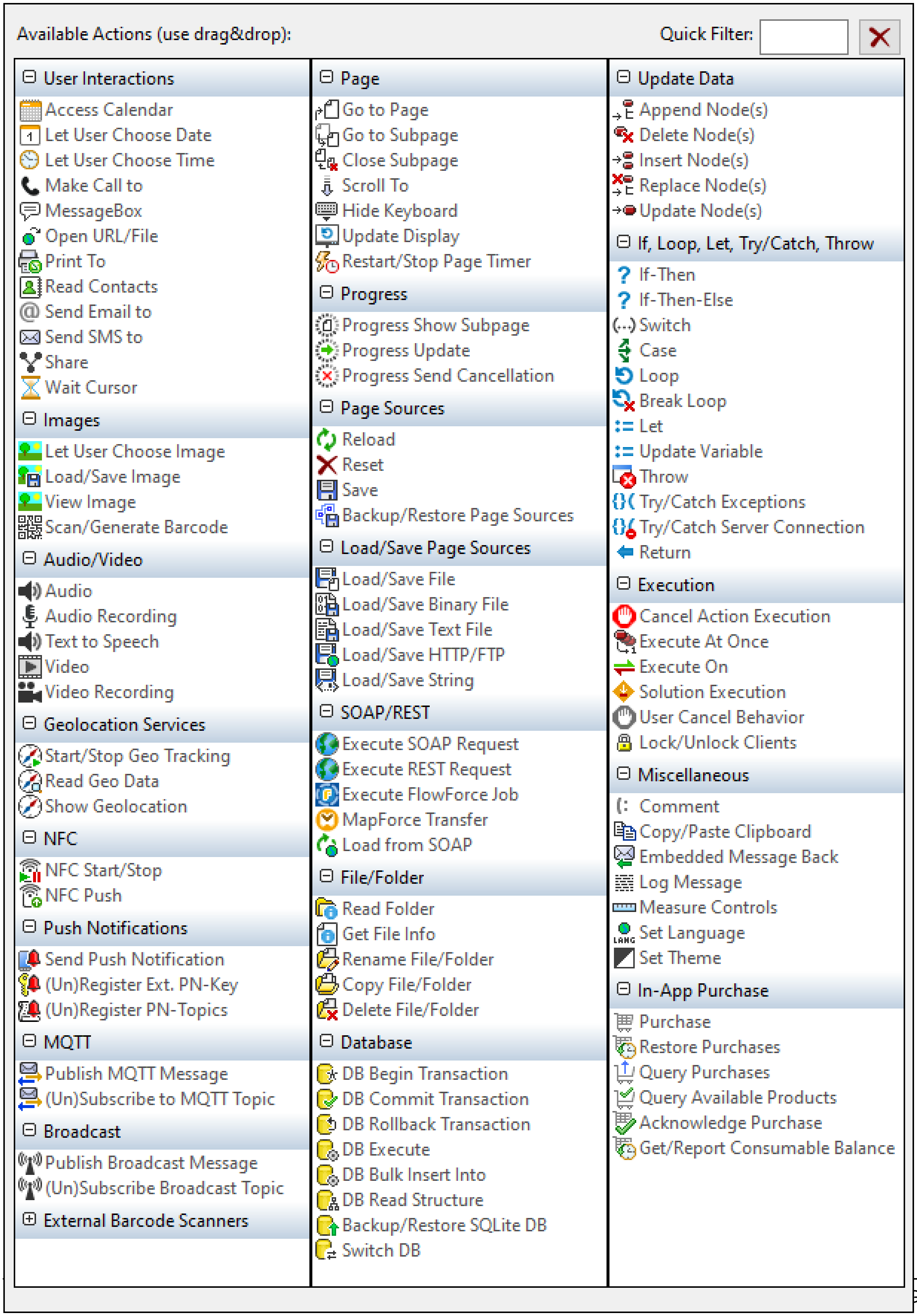
MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-available-db-connection-names()`**
**`mt-db-file-path()`**

# 10.17 Update Data

The following actions are available in the Update Data group of the Actions dialog (*screenshot below*):

- Append Node(s) 875
- Delete Node(s) 879
- Replace Node(s) 883
- Insert Node(s) 880
- Update Node(s)



Available Actions (use drag&drop):

Quick Filter:

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

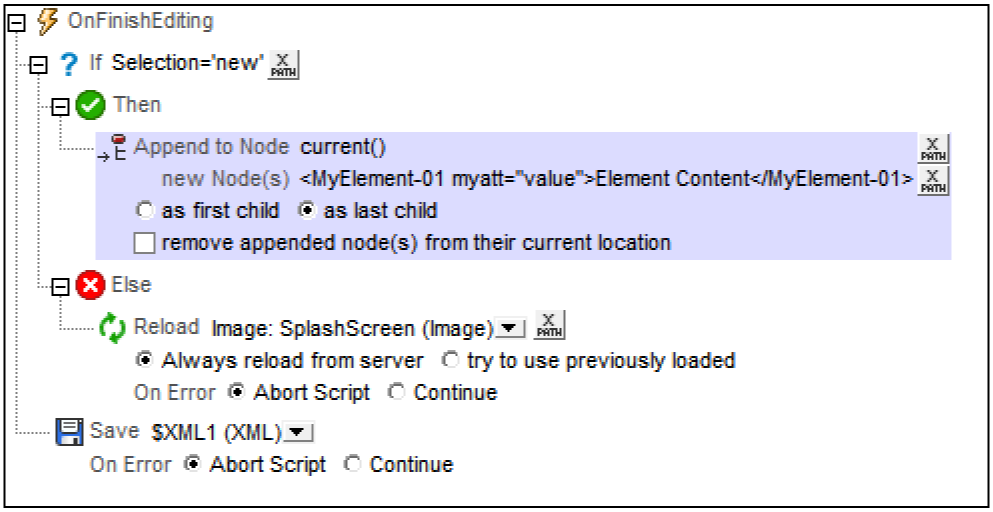
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events* 389 *and* *Control Events* 665.

## 10.17.1 Append Node(s)

The Append Node(s) action appends one or more new nodes as a first or last child (set of nodes) of the node/s selected by the XPath expression for the *Append to Node* setting. The appended node/s can be a single node, sequence of nodes, or an entire tree fragment. These appended nodes are constructed using XQuery's XML constructor syntax. All seven XML node kinds can be constructed using this XQuery syntax: elements, attributes, text, document, comment, processing-instruction, and namespace.



OnFinishEditing
If Selection='new'
Then
Append to Node current()
new Node(s) <MyElement-01 myatt="value">Element Content</MyElement-01>
○ as first child ● as last child
□ remove appended node(s) from their current location
Else
Reload Image: SplashScreen (Image)
● Always reload from server ○ try to use previously loaded
On Error ● Abort Script ○ Continue
Save $XML1 (XML)
On Error ● Abort Script ○ Continue

**Note:** The difference between Insert Node(s) 880 and Append Node(s) 875 is that Insert Node(s) 880 adds the node/s before the selected node/s, whereas Append Node(s) 875 adds the node/s as (first or last) child nodes of the selected node/s.

### Location of appended node/s

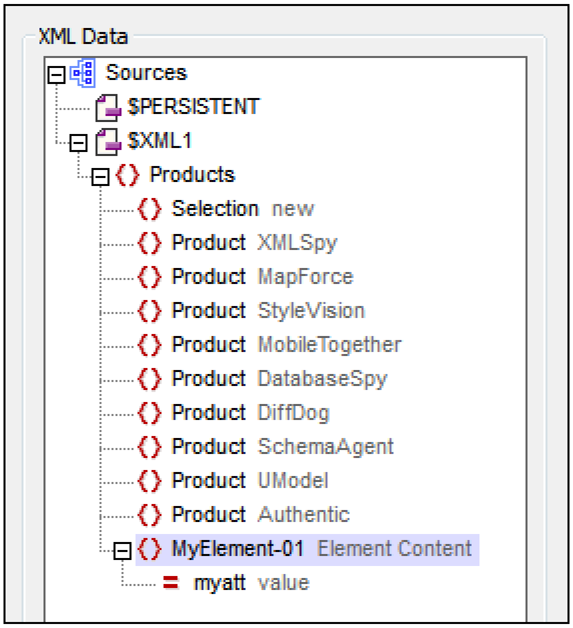
The new node/s are appended as the first or last child node/s of the node/s returned by the XPath expression for this setting (*Append to Node*). In the screenshot above, the new node/s are appended as the last child nodes of the context node, the `Products` element (selected with the XPath expression `current()`). To select whether the new node/s should be appended as the first or last child node/s, select the appropriate radio button in the action's definition.

### New nodes

New nodes can be entered as direct XML constructors as in the screenshot above:

```
<MyElement-01 myatt="value">Element Content</MyElement-01>
```

This appends the element `MyElement-01` after the last `Product` element, as in the screenshot below.



You can also use an XPath locator expression to append a node (and all its descendants) from a page source. For example:

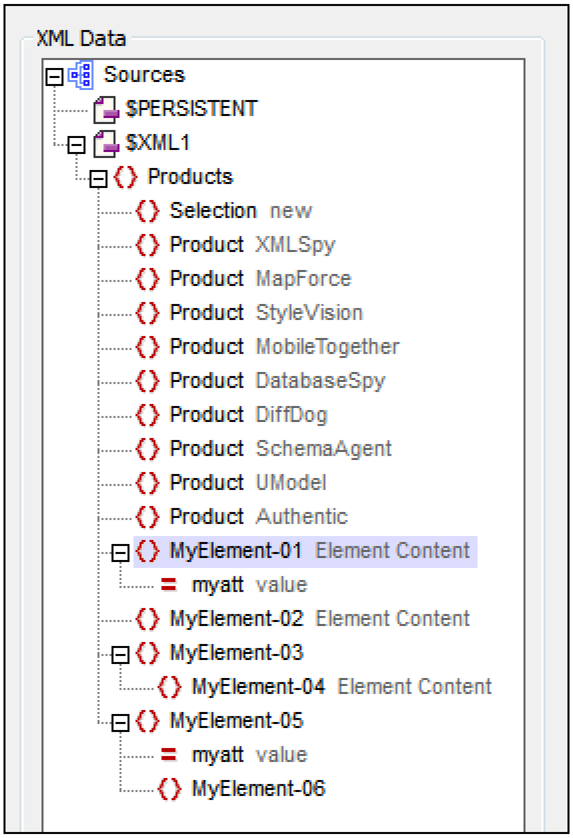
```
$XML2/Row
```

XQuery's computed node constructors can also be used, for example:

```
element MyElement-01 {xs:string("Element Content")}
attribute myatt{"value"}
```

The following XPath expression produces the output shown in the screenshot below, appended as the last child nodes of the `Products` element, that is, after the last current child node (the last `Product` node).

```
<MyElement-01 myatt="value">Element Content</MyElement-01>,
element MyElement-02 {"Element Content"},
element MyElement-03 {element MyElement-04 {"Element Content"}},
element MyElement-05{attribute myatt{"value"}, element MyElement-06{}}
```



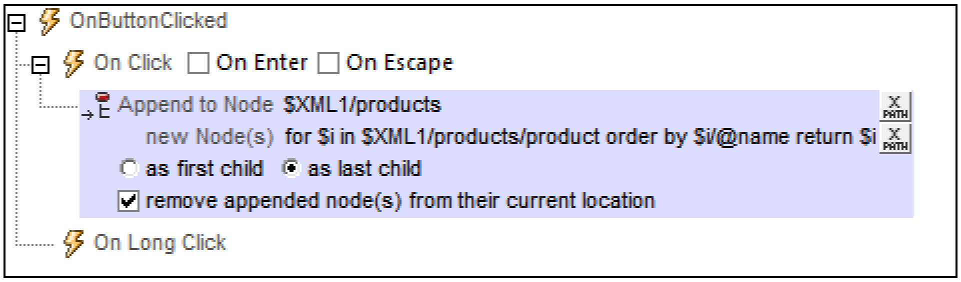
XML Data

```
Sources
  $PERSISTENT
  $XML1
    Products
      Selection new
      Product XMLSpy
      Product MapForce
      Product StyleVision
      Product MobileTogether
      Product DatabaseSpy
      Product DiffDog
      Product SchemaAgent
      Product UModel
      Product Authentic
      MyElement-01 Element Content
        myatt value
      MyElement-02 Element Content
      MyElement-03
        MyElement-04 Element Content
      MyElement-05
        myatt value
        MyElement-06
```

## Removing appended nodes from their original locations

If the appended node/s are obtained from one of the project's page sources, you can delete the node/s from their original location by selecting the *Remove appended node(s) from their current location* check box. If new nodes are constructed directly—that is, without reference to the project's page sources—then selecting this option will have no effect on the page sources.

A good example of how to use the *Remove appended node(s) from their current location* option is the sorting of nodes. Suppose we have a tree with this structure: `$XML1/products/product/@name`. We want to sort the **`product`** nodes on the basis of their **`@name`** values. We can do this with the Append Node(s) definition shown in the screenshot below.



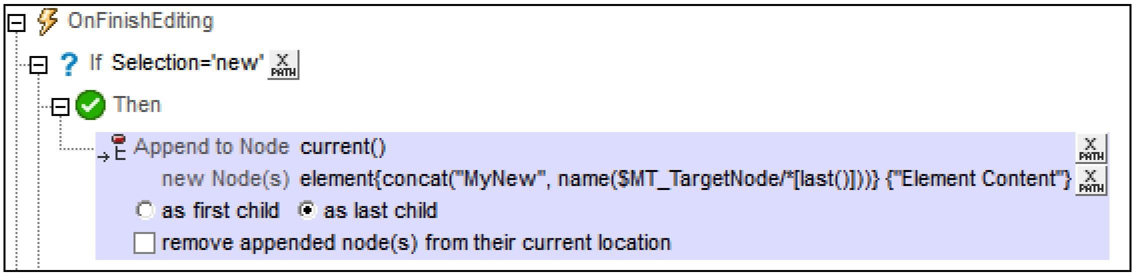
OnButtonClicked
On Click □ On Enter □ On Escape
Append to Node $XML1/products
new Node(s) for $i in $XML1/products/product order by $i/@name return $i
○ as first child ● as last child
☑ remove appended node(s) from their current location
On Long Click

- We append the new nodes as last child to the `$XML1/products` node.

- The new nodes are generated with the XPath expression: **`for $i in $XML1/products/product`** `order by $i/@name` **`return $i`**. The **`order by`** clause sorts the sequence of **`product`** items before iterating over them.
- The *Remove appended node(s) from their current location* option removes the original unordered product sequence. This leaves us with the ordered sequence that was appended.

## The **$MT_TargetNode** variable

The node in the Append Node(s) definition that is targeted as the node in which to append child node/s, is automatically saved in MobileTogether Designer's built-in variable **`$MT_TargetNode`**. This variable can then be used in the second XPath expression of the definition, as has been done in the screenshot below.

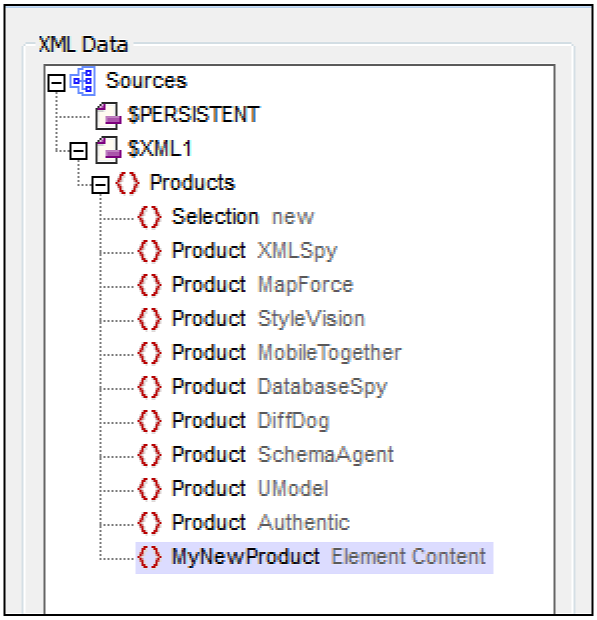


OnFinishEditing
If Selection='new'
Then
Append to Node current()
new Node(s) element{concat("MyNew", name($MT_TargetNode/*[last()]))} {"Element Content"}
○ as first child ● as last child
□ remove appended node(s) from their current location

The second XPath expression uses the target node (`$MT_TargetNode`) to find the target node's last child element and then uses the name of that child element to build the name of the new element.

```
    element {concat("MyNew", name($MT_TargetNode/*[last()]))} {"Element Content"}
```

The result of the Append Node(s) action defined above (when the target node is `$XML1/Products`) is shown in the screenshot below.



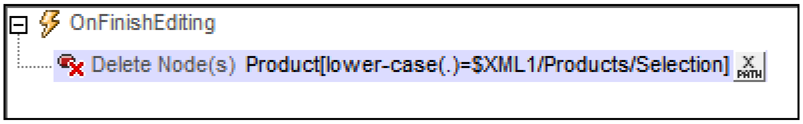
XML Data
Sources
$PERSISTENT
$XML1
Products
Selection new
Product XMLSpy
Product MapForce
Product StyleVision
Product MobileTogether
Product DatabaseSpy
Product DiffDog
Product SchemaAgent
Product UModel
Product Authentic
MyNewProduct Element Content

## MobileTogether extension functions

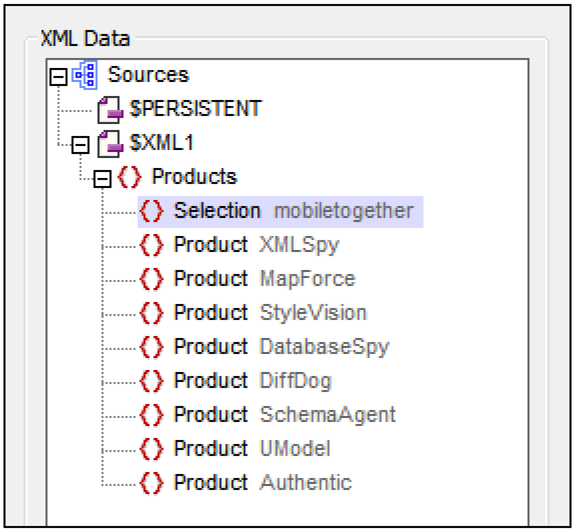
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.17.2 Delete Node(s)

The Delete Node(s) action deletes the node/s selected in the action's XPath expression.


OnFinishEditing
Delete Node(s) Product[lower-case(.)=$XML1/Products/Selection]
X PATH

The definition in the screenshot above is for an `OnFinishedEditing` event of a combo box that updates the `$XML1Products/Selection` element. The context node is `Products`. The action deletes the child `Product` element that has content, which when converted to lowercase matches the (lowercase) content of the `Selection` element. In the screenshot below, `mobiletogether` has been selected in the combo box and becomes the value of the Selection element. The `Product` element containing the text `"MobileTogether"` has been deleted.



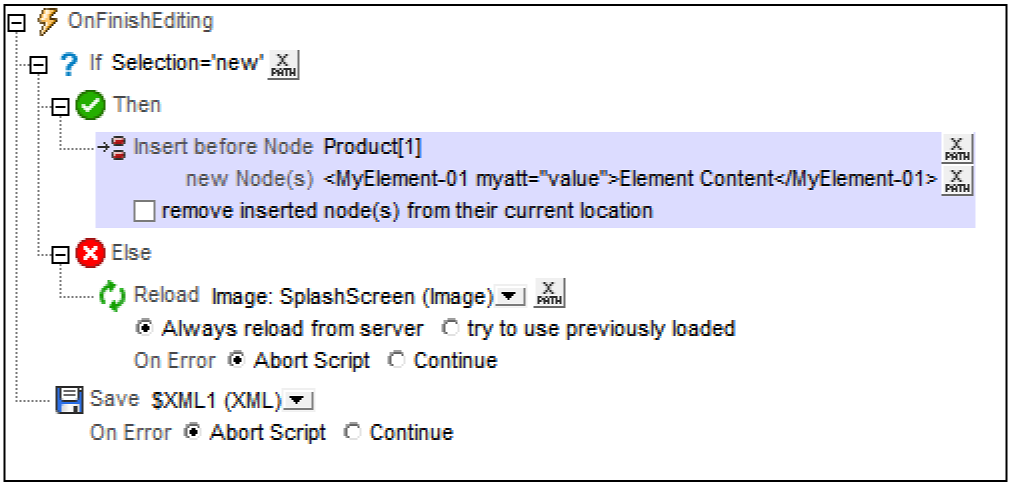
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a

full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.17.3 Insert Node(s)

The Insert Node(s) action inserts one or more new nodes before the node/s selected by the XPath expression for the *Insert before Node* setting. The inserted node/s can be a single node, sequence of nodes, or an entire tree fragment. These inserted nodes are constructed using XQuery's XML constructor syntax. All seven XML node kinds can be constructed using this XQuery syntax: elements, attributes, text, document, comment, processing-instruction, and namespace.



```
OnFinishEditing
  If Selection='new'
    Then
      Insert before Node Product[1]
             new Node(s) <MyElement-01 myatt="value">Element Content</MyElement-01>
      □ remove inserted node(s) from their current location
    Else
      Reload Image: SplashScreen (Image)
      ● Always reload from server  ○ try to use previously loaded
      On Error ● Abort Script  ○ Continue
  Save $XML1 (XML)
  On Error ● Abort Script  ○ Continue
```

**Note:** The difference between Insert Node(s)[880] and Append Node(s)[875] is that Insert Node(s)[880] adds the node/s before the selected node/s, whereas Append Node(s)[875] adds the node/s as (first or last) child nodes of the selected node/s.

### Location of inserted node/s

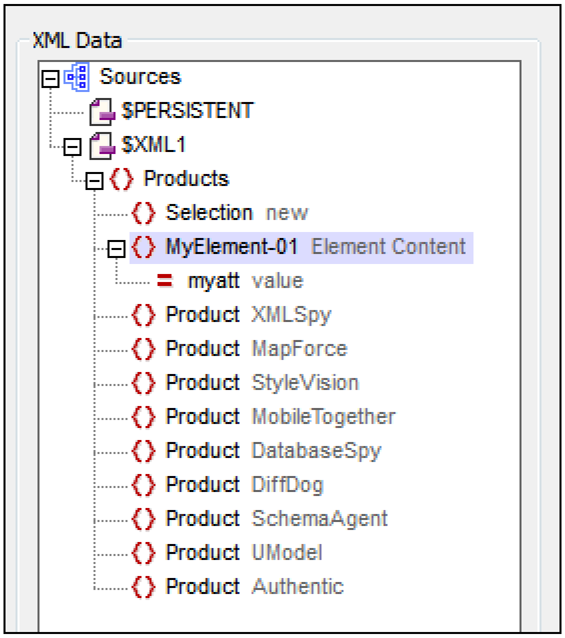
The new node/s are inserted before the node/s returned by the XPath expression for this setting (*Insert before Node*). In the screenshot above, the new node/s are inserted before the first `Product` element (selected with the XPath expression `Product[1]`). The context node is the parent of `Product`, a node named `Products`. If the predicate `[1]` is not used, all the `Product` children of `Products` will be returned by the XPath expression, and the new node/s will be inserted before each `Product` element.

### New nodes

New nodes can be entered as direct XML constructors as in the screenshot above:

```
<MyElement-01 myatt="value">Element Content</MyElement-01>
```

This inserts the element `MyElement-01` before the first `Product` element, as in the screenshot below.



```
Sources
    $PERSISTENT
    $XML1
        Products
            Selection new
            MyElement-01 Element Content
                myatt value
            Product XMLSpy
            Product MapForce
            Product StyleVision
            Product MobileTogether
            Product DatabaseSpy
            Product DiffDog
            Product SchemaAgent
            Product UModel
            Product Authentic
```

You can also use an XPath locator expression to insert a node (and all its descendants) from a page source on the page. For example:

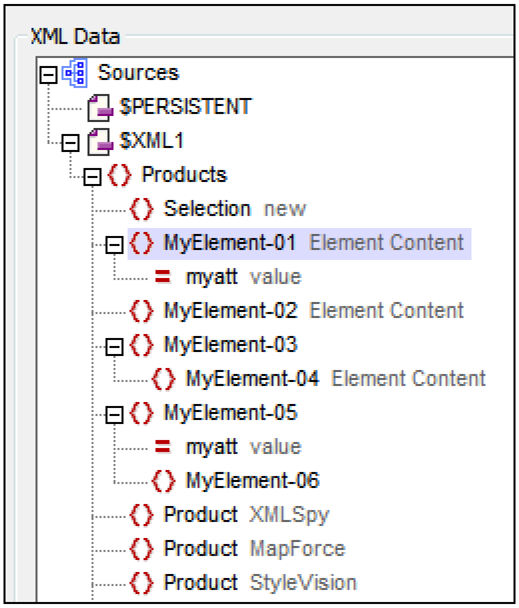
```
$XML2/Row
```

XQuery's computed node constructors can also be used, for example:

```
element MyElement-01 {xs:string("Element Content")}
attribute myatt{"value"}
```

The XPath expression below produces the output shown in the screenshot below, inserted above the first `Product` element.

```
<MyElement-01 myatt="value">Element Content</MyElement-01>,
element MyElement-02 {"Element Content"},
element MyElement-03 {element MyElement-04 {"Element Content"}},
element MyElement-05{attribute myatt{"value"}, element MyElement-06{}}
```



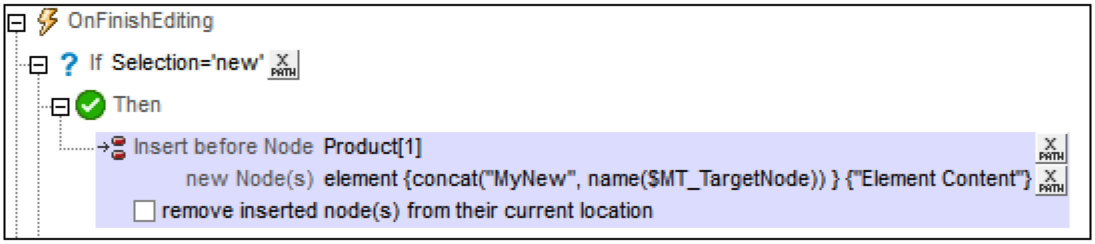
XML Data
Sources
$PERSISTENT
$XML1
Products
Selection new
MyElement-01 Element Content
myatt value
MyElement-02 Element Content
MyElement-03
MyElement-04 Element Content
MyElement-05
myatt value
MyElement-06
Product XMLSpy
Product MapForce
Product StyleVision

## Removing inserted nodes from their original locations

If the inserted node/s are obtained from one of the project's page sources, you can delete the node/s from their original location by selecting the *Remove inserted node(s) from their current location* check box. If new nodes are constructed directly—that is, without reference to the project's page sources—then selecting this option will have no effect on the page sources.

## The **$MT_TargetNode** variable

In the Insert Node(s) definition, the node before which the insertion is carried out is automatically saved in MobileTogether Designer's built-in variable **`$MT_TargetNode`**. This variable can then be used in the second XPath expression of the definition, as has been done in the screenshot below.

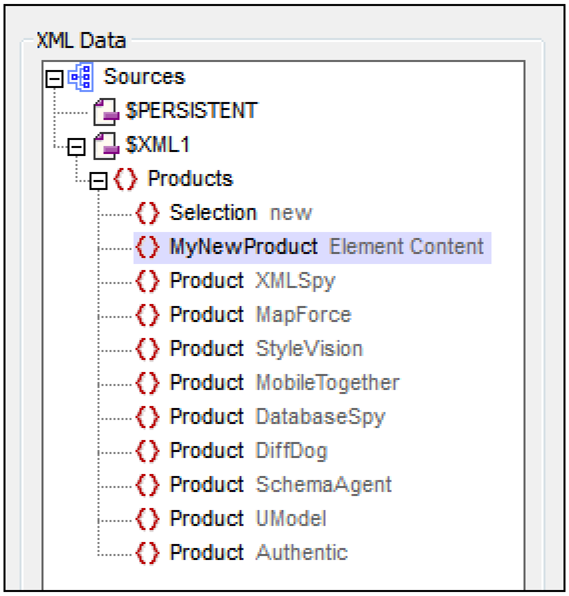


OnFinishEditing
If Selection='new'
Then
Insert before Node Product[1]
new Node(s) element {concat("MyNew", name($MT_TargetNode)) } {"Element Content"}
□ remove inserted node(s) from their current location

The second XPath expression that creates the new node using the same name as the target node (`$MT_TargetNode`) as part of the new node's name.

```
    element {concat("MyNew", name($MT_TargetNode))} {"Element Content"}
```

The result of the Insert Node(s) action defined above is shown in the screenshot below.



XML Data

Sources
$PERSISTENT
$XML1
Products
Selection new
MyNewProduct Element Content
Product XMLSpy
Product MapForce
Product StyleVision
Product MobileTogether
Product DatabaseSpy
Product DiffDog
Product SchemaAgent
Product UModel
Product Authentic

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.17.4 Replace Node(s)

When the action is triggered, the following happens:

1. The **nodeset** specified by the *Subnode(s)* XPath expression is deleted from the **target node** (specified in the *Target Node* setting), if it exists there.
2. The **nodeset** specified by the *Subnode(s)* XPath expression is copied from a **source node** (specified in the *Parent of source node(s)* setting) to the target node. The nodeset can be appended either as a first child or last child of the target node.

The XPath expression of the *Subnode(s)* setting is evaluated separately for source and target: in the context, respectively, of the node specified as the source node and the node specified as the target node.

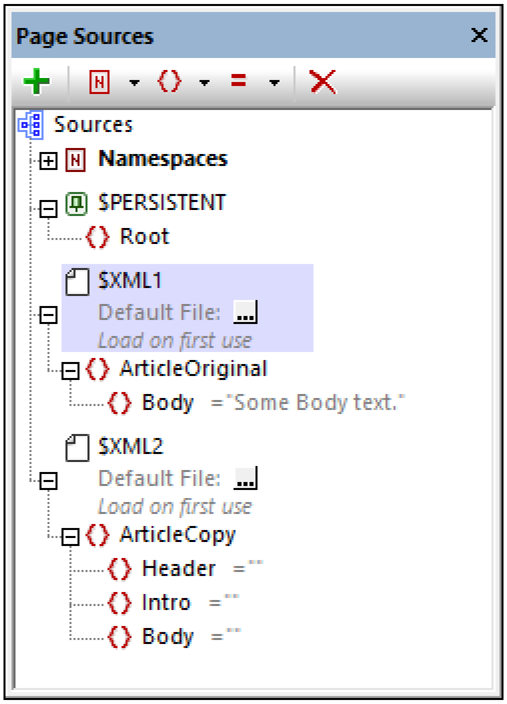
The Replace Node(s) action is in effect a combination of the two actions Delete Node(s) 879 and Append Node(s) 875.

### Examples

Given below are two examples that demonstrate how the Replace Node(s) action works.

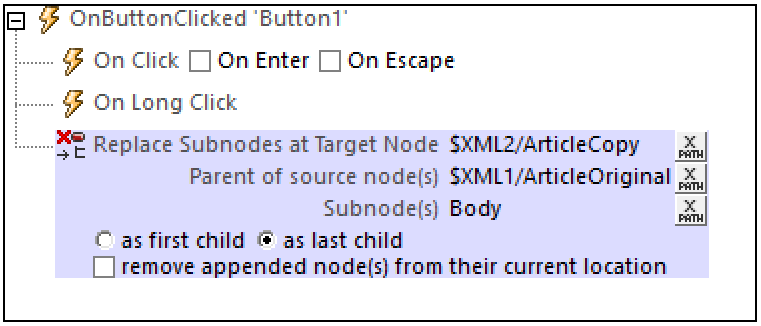
*Simple replacement*
Consider the structure of the two XML trees shown in the Page Sources pane shown below.

- **`$XML1/ArticleOriginal`** has a child element named **`Body`**.
- **`$XML2/ArticleCopy`** has child elements named **`Header`**, **`Intro`**, and **`Body`**, in that order



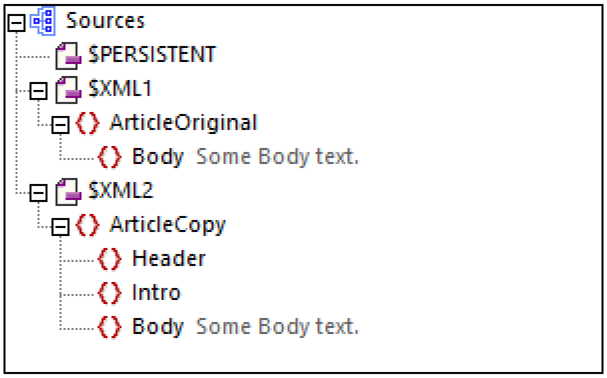
```
Sources
  ⊞ Namespaces
  ⊟ $PERSISTENT
      {} Root
  ⊟ $XML1
      Default File: ...
      Load on first use
    ⊟ {} ArticleOriginal
        {} Body  ="Some Body text."
  ⊟ $XML2
      Default File: ...
      Load on first use
    ⊟ {} ArticleCopy
        {} Header  =""
        {} Intro  =""
        {} Body  =""
```

We can replace the **`Body`** element of **`ArticleCopy`** with the **`Body`** element of ArticleOriginal by using the Replace Node action as shown in the screenshot below.



```
⊟ OnButtonClicked 'Button1'
    On Click □ On Enter □ On Escape
    On Long Click
    Replace Subnodes at Target Node $XML2/ArticleCopy
             Parent of source node(s) $XML1/ArticleOriginal
                          Subnode(s) Body
    ○ as first child ● as last child
    □ remove appended node(s) from their current location
```

The action appends a *subnode* named **`Body`** as a last child from a *source node* named **`ArticleOriginal`** to a target named **`ArticleCopy`**. The screenshot below shows the page sources during a simulation, after the action has been executed. Notice that the **`Body`** element of **`ArticleCopy`** has been replaced by the **`Body`** element of **`ArticleOriginal`**.



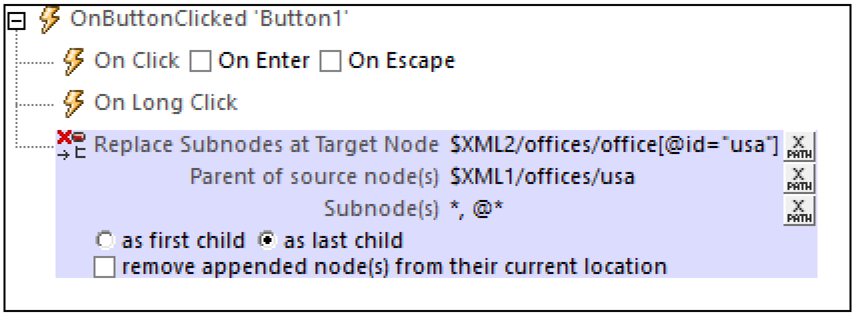
Sources
$PERSISTENT
$XML1
ArticleOriginal
Body Some Body text.
$XML2
ArticleCopy
Header
Intro
Body Some Body text.

Notice that **`Header`** and **`Intro`** have not been touched, but **`Body`** has been replaced. If **`ArticleCopy`** had had no **`Body`** element, then a new **`Body`** element would have been appended.

The XPath expression **`Body`** (specified in the *Subnode(s)* setting) is evaluated in the context of the *Source node* and *Target node* XPath expression, respectively, in order to (i) locate the source nodeset to copy, and (ii) locate the node from which to delete and into which to append.

*Replacement of an entire subtree*

The screenshot below shows the settings required to append an entire nodeset from one (source) node to another (target) node. All the descendant elements of the source node **`$XML1/office/usa`**, together with their attributes, are appended to the target node **`$XML2/offices/office[@id="usa"]`**.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Replace Subnodes at Target Node $XML2/offices/office[@id="usa"]
Parent of source node(s) $XML1/offices/usa
Subnode(s) *, @*
○ as first child ● as last child
□ remove appended node(s) from their current location

The exact sequence of actions is as follows:

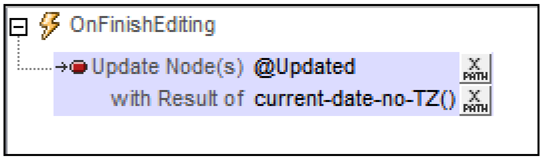
1. The *Subnode(s)* XPath expression `*, @*` locates all descendant elements of the target node **`$XML2/offices/office[@id="usa"]`**, and **deletes** them.
2. The *Subnode(s)* XPath expression `*, @*` locates all descendant elements of the source node **`$XML1/office/usa`**, and copies them as a single nodeset body to the target node **`$XML2/offices/office[@id="usa"]`**. Since all of the target node's child elements have been deleted, the new subnode will entirely replace the previous descendants of the target node.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.17.5 Update Node(s)

The Update Node(s) action updates one or more specified nodes with the specified value. Both the update node and update value are specified with XPath expressions. In the screenshot below, the `@Updated` attribute node of the XPath context element is updated with the current date (the result of evaluating the XPath function `current-date-no-TZ()`).



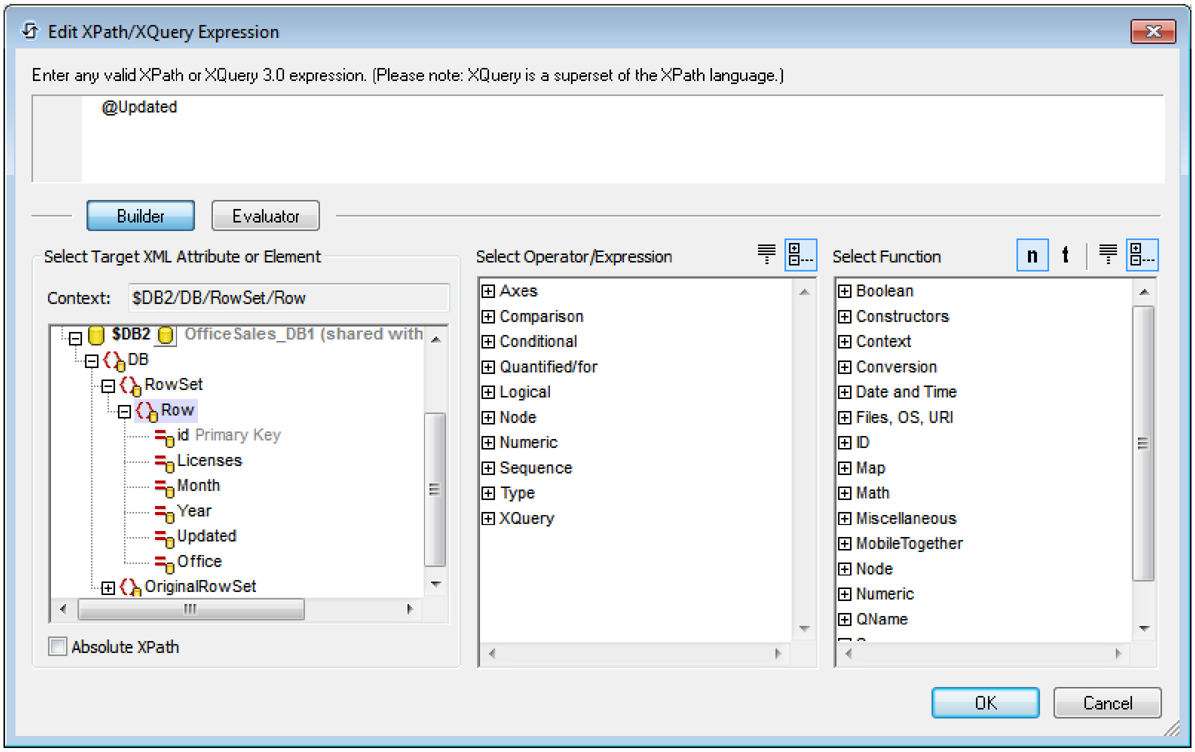
There are a few points to note when specifying the update node/s and update value/s:

- Importance of context node for relative paths. (*See below*[886] *for details*).
- The target node for the update can be referenced with the **`$MT_TargetNode`**[1304] variable. (*See below*[888] *for details*).
- If the source node is an element with mixed content (text and elements), then only the text content of the mixed-content element is used for the update. The text content of descendant elements is ignored. (*See below*[889] *for details*).
- Multiple target nodes can be specified via an array. (*See below*[890] *for details*).

### Importance of context node for relative paths

If the node to be updated is specified as a relative path, then only those nodes are updated that are descendants of the context node. To update a descendant node of sibling elements, an absolute locator path will need to be used. This is explained in more detail below.

- In the screenshot below, notice that the context node (indicated in the *Context* field) is the `Row` element of the DB. The context node is the node within which the control (for which the action is being defined) is located (or with which the control is associated).
- The `@Updated` attribute that will be updated is therefore the `@Updated` attribute of that particular `Row` element. What this means is that when a `Row` element's control event is triggered, then the `@Updated` attribute of **only that `Row`** element is updated, in this case with the current date.



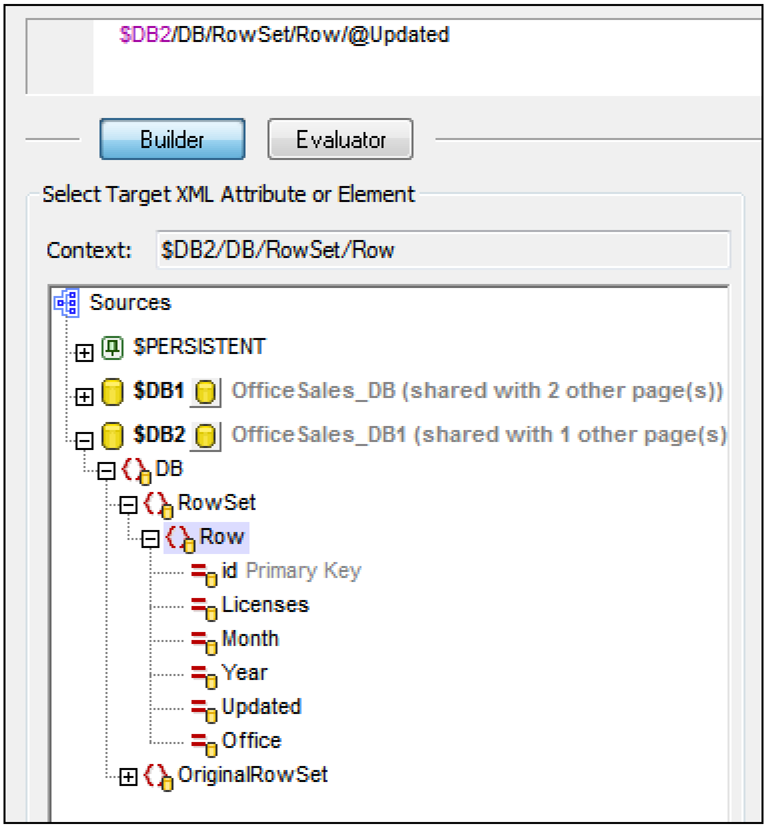
Edit XPath/XQuery Expression

Enter any valid XPath or XQuery 3.0 expression. (Please note: XQuery is a superset of the XPath language.)

@Updated

Context: $DB2/DB/RowSet/Row

- If the XPath expression were changed so that it addressed the `@Updated` node starting from the root node (like this, and as in the screenshot below: `$DB2/DB/RowSet/Row/@Updated`), then the `@Updated` node of **all** `Row` elements would be updated.



$DB2/DB/RowSet/Row/@Updated

Builder
Evaluator
Select Target XML Attribute or Element

Context: $DB2/DB/RowSet/Row

Sources
$PERSISTENT
$DB1 OfficeSales_DB (shared with 2 other page(s))
$DB2 OfficeSales_DB1 (shared with 1 other page(s)
DB
RowSet
Row
id Primary Key
Licenses
Month
Year
Updated
Office
OriginalRowSet

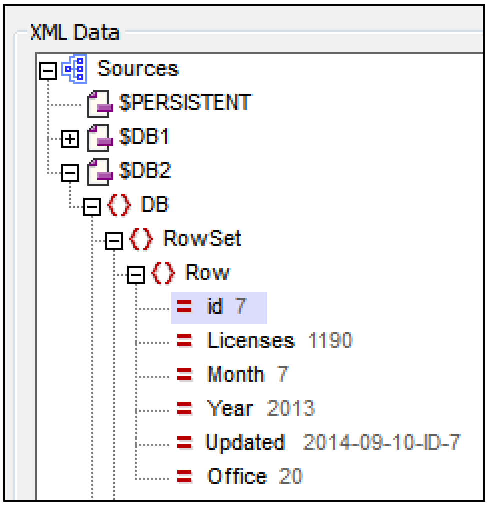
- Notice from the screenshot above, that you have access via XPath expressions to all the nodes of all page sources.

## Referencing the specified target node with $MT_TargetNode

After a target node for the update is defined, you can reference this node with the variable **`$MT_TargetNode`**. For example:

```
Update Node : @Updated
Update Value: concat(current-date-no-TZ(), '-ID-', $MT_TargetNode/../@id)
```

would give an update value that would have the `@id` attribute value of the current `Row` element suffixed to the current date, as in the screenshot below.



XML Data
Sources
$PERSISTENT
$DB1
$DB2
DB
RowSet
Row
id 7
Licenses 1190
Month 7
Year 2013
Updated 2014-09-10-ID-7
Office 20

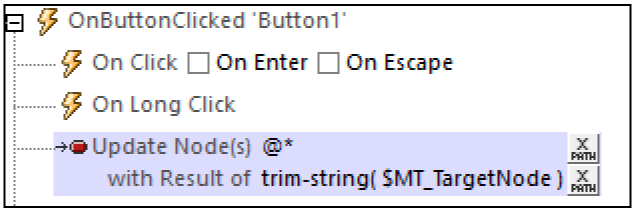
Now, if the update nodes were the `@Updated` attributes of all `Row` elements, and the XPath expression for the update value were the same as in the previous example:

```
Update Node : $DB2/DB/RowSet/Row/@Updated
Update Value: concat(current-date-no-TZ(), '-ID-', $MT_TargetNode/../@id)
```

then the `@Updated` attribute of each `Row` element would have a value that would have the `@id` attribute value of its own `Row` element suffixed to the current date.

## Modifying and updating nodes with $MT_TargetNode

The **`$MT_TargetNode`** variable can be very useful if you want to modify and update nodes in one joint action. An example is shown in the screenshot below.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Update Node(s) @*
with Result of trim-string( $MT_TargetNode )

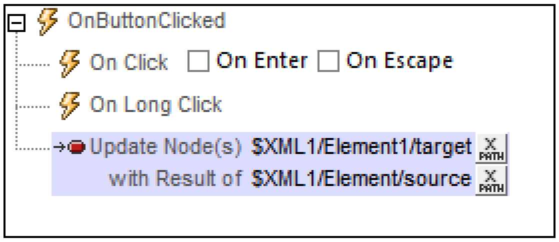
The action in the screenshot works as follows:

- The nodes to update are all the attributes of the current element. These nodes are specified by the XPath expression of *Update Node(s)*: `@*`
- As each attribute is processed it becomes the subject of **`$MT_TargetNode`**
- The *Result* setting specifies the updated value of the node, which, in the example above, is the original value of the attribute (obtained via **`$MT_TargetNode`**)—modified in that its leading and trailing spaces are removed by the **`trim-string`** XPath function.
- The overall result is that each attribute will have any leading or trailing spaces removed

## Source elements with mixed content

If an element with mixed content (text and element/s) is located with an XPath locator expression, then the text content of the mixed-content element only is returned. The text content of descendant elements is ignored.

This is best explained with an example of the Update Node(s) action 886. Consider the Update Node(s) action 886 defined in the screenshot below.



OnButtonClicked
On Click □ On Enter □ On Escape
On Long Click
Update Node(s) $XML1/Element1/target
with Result of $XML1/Element/source

If the XML tree had the following structure and content:

```
<Element1>
   <source>AAA
      <subsource>BBB</subsource>
  </source>
  <target></target>
</Element1>
```

Then the `target` element would be updated with the text content of the mixed-content element `source`, while ignoring the content of its child element `subsource`. The node named `target` will be updated to `<target>AAA</target>`.

**Note:** If you wish to include the text content of descendant node/s, use a **`string`** function. Using the XML example above, for instance, the expression `string($XML1/Element1/source, '')` will return `"AAABBB"`.

**Note:** Charts use the XPath compliant method of serializing: When a mixed-content element is located using an XPath locator expression, then the text content of descendant elements is also serialized.

## Updating multiple nodes via XPath arrays

You can update multiple nodes (the target nodes) by locating these nodes via an XPath array. The update values must then also be submitted as an array, one which has the same size as the target-node array. The update happens 1:1 and member-wise: the first member of the value array updates the first member of the target array, and so on. For example:

```
Update Node : $XML1/User/Message/[@subject, @date, @senderID]
Update Value: ['Monthly Meeting June', '2018-05-31', 3485]
```

Note the following points:

- The **`$MT_TargetNode`** variable is set to the array that identifies the target nodes. You can check whether the variable contains an array as follows: **`if($MT_TargetNode instance of array(*)) then...`**
- The evaluation and mapping of the arrays are carried out first; the assignments are executed as a group subsequently. This means that the value of a target node can be used as an input.
- Each member of the target array is handled separately. As a result, even if one member of the target array produces an error (either itself or while its value was evaluated), the updates of other target nodes will continue. Appropriate error messages, however, are generated.

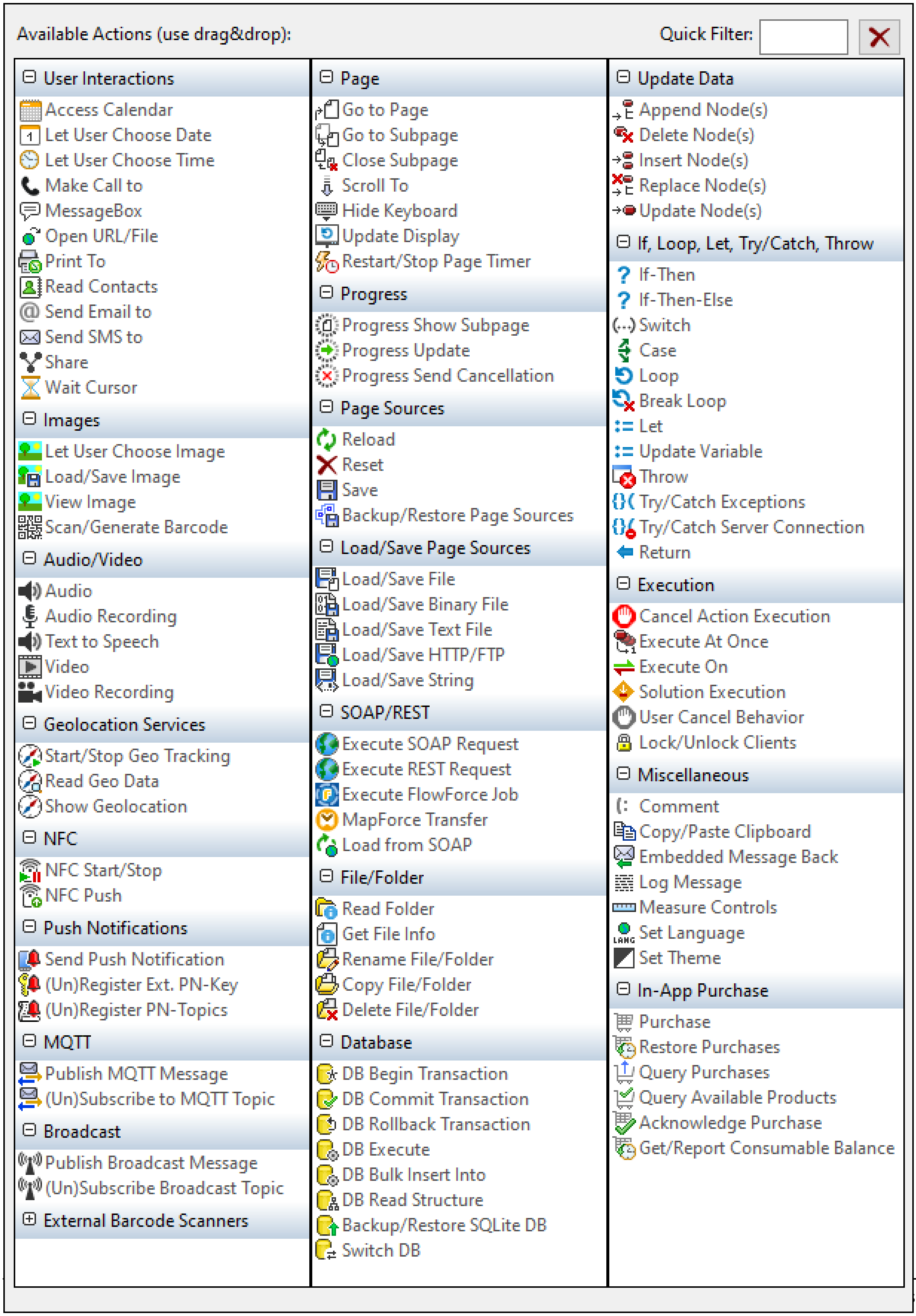
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.18 If, Loop, Let, Try/Catch, Throw

The following actions are available in the If, Loop group of the Actions dialog (*screenshot below*):

- If-Then 894
- If-Then-Else 894
- Switch, Case 895
- Loop 897
- Break Loop 899
- Let 900
- Update Variable 903
- Throw 905
- Try/Catch Exceptions 907
- Try/Catch Server Connection 908
- Return



Available Actions (use drag&drop):

Quick Filter:

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

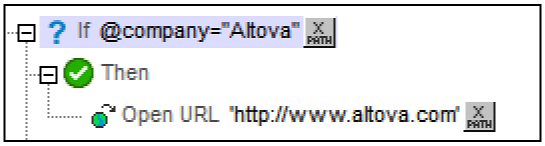
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events*[389] *and Control Events*[665].

## 10.18.1 If-Then

Sets the action to be carried out if the `IF` condition evaluates to `true`. The action to be carried out is dropped as a child of the `THEN` clause.



If @company="Altova"
Then
Open URL 'http://www.altova.com'

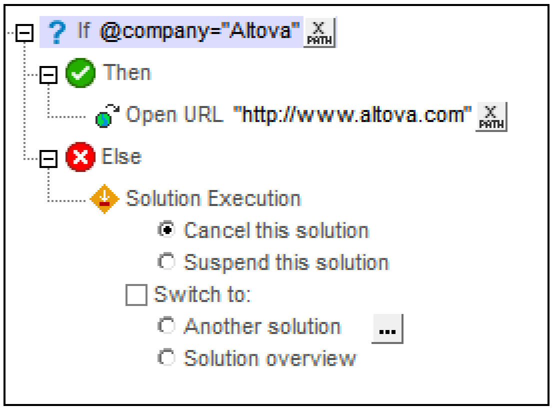
The condition in the definition above tests whether the current element has an attribute called `company` that has a value of `Altova`. If `true`, the Altova website URL is opened.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.18.2 If-Then-Else

Sets the actions to be carried out for both the `true` and `false` evaluations of the `IF` condition. The action to be carried out for a `true` evaluation is dropped as a child of the `THEN` clause. The action to be carried out for a `false` evaluation is dropped as a child of the `ELSE` clause.



If @company="Altova"
Then
Open URL "http://www.altova.com"
Else
Solution Execution
● Cancel this solution
○ Suspend this solution
□ Switch to:
○ Another solution ...
○ Solution overview

The condition in the definition above tests whether the current element has an attribute called `company` that has a value of `Altova`. If `true`, the Altova website URL is opened. If `false`, the solution is exited.

### MobileTogether extension functions

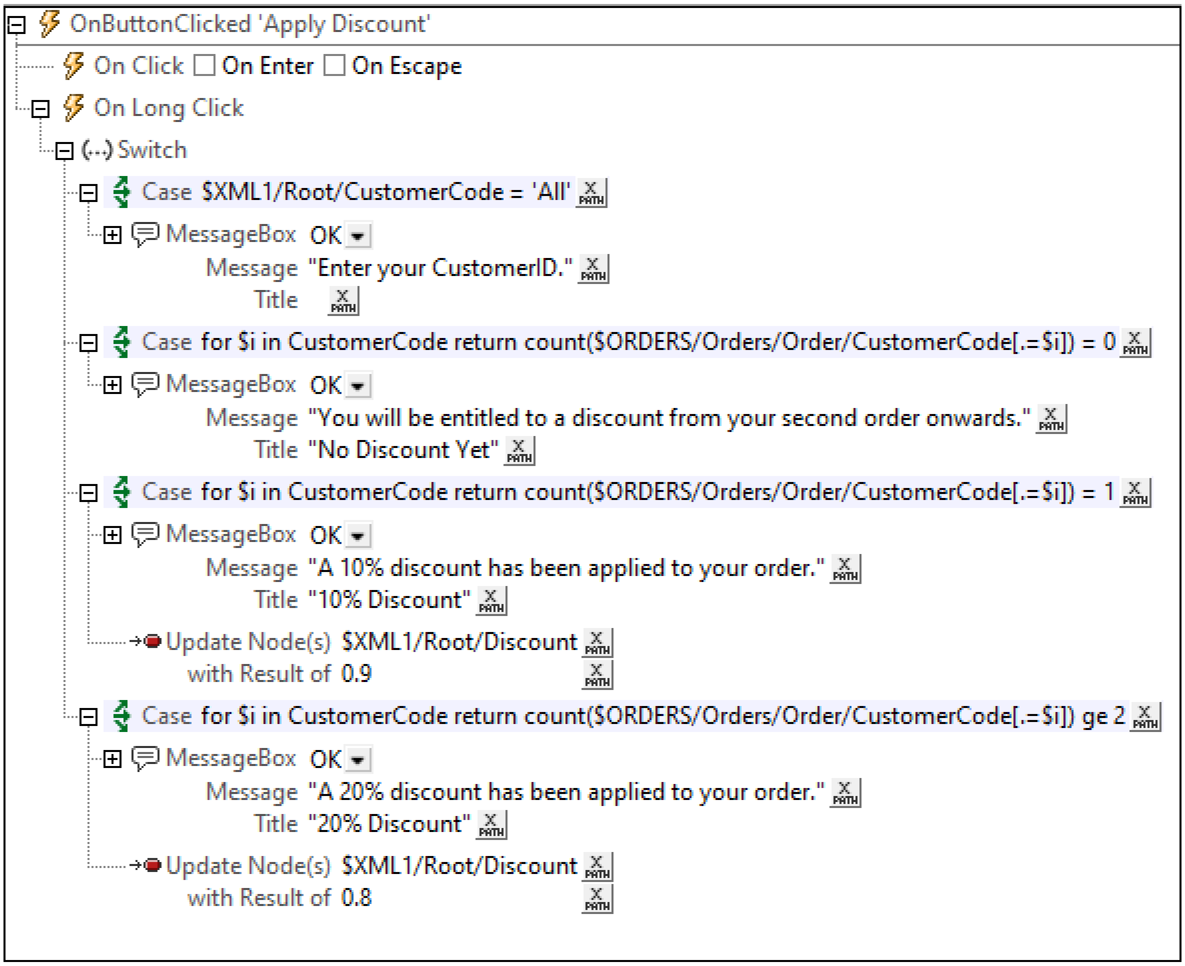
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.18.3 Switch, Case

The Switch and Case actions work together, with a Switch action containing one or more Case actions (*see screenshot below*). You can think of the Switch–Case-action mechanism as follows:

- Each Case action defines within it an alternative set of actions to execute. A Case action will be triggered if the trigger condition defined for it evaluates to `true`.
- The Switch action executes the first Case action (in the sequence of Case actions) that evaluates to `true`. After the triggered Case action is executed, the Switch action ends: no subsequent Case action is evaluated.
- When a Case action is executed, the action tree defined within it is executed.

To add a Case action to a Switch action, drag the Case action from the left-hand Actions pane into the right-hand pane and place it at the position where you want it in the sequence of Case actions. For each Case action, drag-and-drop into it whatever actions you want for its action tree.



```
OnButtonClicked 'Apply Discount'
    On Click □ On Enter □ On Escape
    On Long Click
        (...) Switch
            Case $XML1/Root/CustomerCode = 'All'
                MessageBox OK
                    Message "Enter your CustomerID."
                    Title
            Case for $i in CustomerCode return count($ORDERS/Orders/Order/CustomerCode[.=$i]) = 0
                MessageBox OK
                    Message "You will be entitled to a discount from your second order onwards."
                    Title "No Discount Yet"
            Case for $i in CustomerCode return count($ORDERS/Orders/Order/CustomerCode[.=$i]) = 1
                MessageBox OK
                    Message "A 10% discount has been applied to your order."
                    Title "10% Discount"
                Update Node(s) $XML1/Root/Discount
                    with Result of 0.9
            Case for $i in CustomerCode return count($ORDERS/Orders/Order/CustomerCode[.=$i]) ge 2
                MessageBox OK
                    Message "A 20% discount has been applied to your order."
                    Title "20% Discount"
                Update Node(s) $XML1/Root/Discount
                    with Result of 0.8
```

In the screenshot above, the Switch action contains four Case actions (which are highlighted pale blue):

- The first Case action tests whether the content of the `$XML1/Root/CustomerCode` node is the string `All`.
- The next three Case actions test the number of orders previously placed by the current customer (identified by its customer code): that is, whether the number of previous orders is 0, 1, or greater/equal 2.

Each Case action has a set of appropriate actions defined within it. For example: The first Case action covers the situation in our hypothetical solution in which no customer code has been entered. The second Case action defines the actions to perform when no previous order exists in the database for the current customer (in this case, no discount is applied). The third and fourth Case actions define the actions to perform in the event that the current customer has, respectively, one previous order and two or more previous orders in the database; different discounts are applied in each case (10% and 20%, respectively).

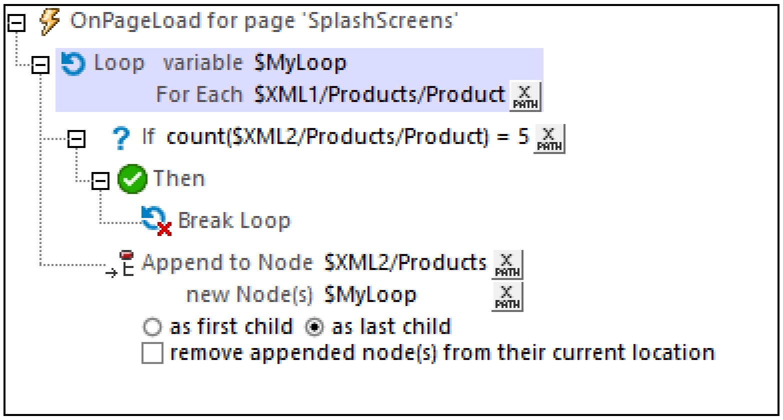
When the Switch action is triggered, the condition of each Case action is tested in turn till one is found that returns `true`. When this happens, the action tree of that Case action is executed and, on its completion, the Switch action ends.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.18.4 Loop

The Loop action (*see screenshot below*) iterates over a sequence of items that you define using the *For Each* setting. Within the loop you can then define a set of actions to perform during every iteration. For example, in the screenshot below, for each iteration, an Append to Node 875 action is carried out. Also inside the loop, a Break Loop 899 action is specified if a certain condition is fulfilled.



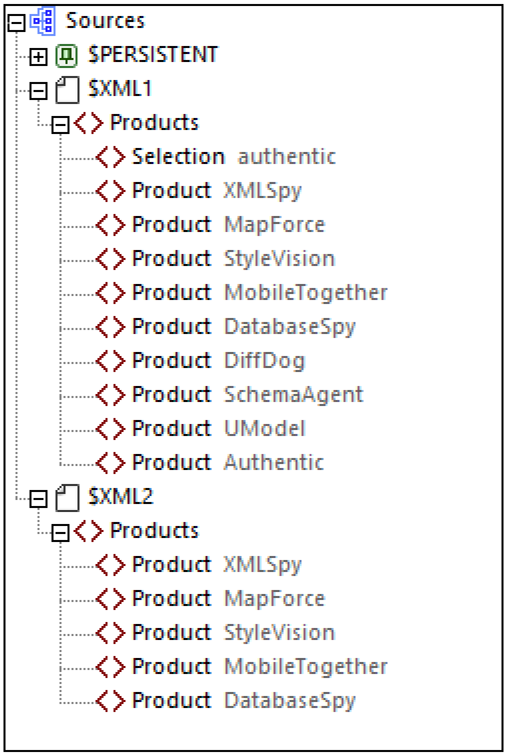
OnPageLoad for page 'SplashScreens'
Loop variable $MyLoop
For Each $XML1/Products/Product
If count($XML2/Products/Product) = 5
Then
Break Loop
Append to Node $XML2/Products
new Node(s) $MyLoop
○ as first child ● as last child
□ remove appended node(s) from their current location

The sequence over which the loop iterates is defined by the XPath expressions of the *For Each* setting. The key points to note are given below.

- *For Each:* Can be a sequence named in the XPath expression (for example: `1 to 7`) or one obtained from an XML tree (for example: `$XML1/Products/Product` selects a sequence of all `Product` elements in the `$XML1` tree; *see screenshot above*). If the loop does not contain a Break Loop 899 action, then the loop is exited when all iterations have been completed.
- *Loop variable:* The loop variable is the variable that holds the item of the sequence that is currently being iterated over. The loop variable is identified by a name, which you enter by first double-clicking after the `$` sign and then typing the name. In the screenshot above, the loop variable is named **`MyLoop`**. It is referenced like any other XPath variable, that is, with a **`$`** sign before its name (**`$MyLoop`**). The variable will be in scope within the loop; this means that you cannot reference the variable in an XPath expression that is outside the loop. In the screenshot above, the loop variable is referenced in the *New Node* setting of the Append to Node 875 action. This is a valid reference since the Append to Node 875 action has been created within the loop; the variable is therefore in scope at this point. The **`$MyLoop`** variable shown in the screenshot above will contain the **`Product`** node that is currently being iterated over.

***Note:*** If the loop variable uses nodes from a page source tree, then this tree is locked and cannot be modified by the actions inside the Loop action. In our example above, the tree $XML1 will be locked while the loop is being processed. The modifications are happening in another page source tree (`$XML2`): the new nodes that are added with the Append to Node[875] action are added to `$XML2`. If you want to modify the tree being iterated over, you can do this as follows: Instead of iterating directly over the nodes of the tree, iterate over a number sequence that is tied to the node sequence in the tree. For example, instead of iterating over the sequence of `Product` nodes in the screenshot example above, we can iterate over a range of numbers that is tied to the node sequence. The XPath expression of the *For Each* setting can be changed from `$XML1/Products/Product` to `for $i in 1 to count($XML1/Products/Product) return $i`. It is a sequence of numbers that is now being iterated over. (The current `Product` node within the loop can be accessed with the XPath expression: `$XML1/Products/Product[$i]`).

The actions defined in the screenshot above duplicates the first five **`Product`** elements in another tree. When the page is loaded, the Loop action iterates over the `$XML1/Products/Product` elements. During each iteration, the current `Product` node is stored in the `MyLoop` variable. This `Product` node (in the `$MyLoop` variable) is then added as the last child of the `$XML2/Products` node. The loop continues till the fifth `Product` element has been copied from `$XML1/Products` to `$XML2/Products`. *See screenshot below.*



```
Sources
  $PERSISTENT
  $XML1
    Products
      Selection authentic
      Product XMLSpy
      Product MapForce
      Product StyleVision
      Product MobileTogether
      Product DatabaseSpy
      Product DiffDog
      Product SchemaAgent
      Product UModel
      Product Authentic
  $XML2
    Products
      Product XMLSpy
      Product MapForce
      Product StyleVision
      Product MobileTogether
      Product DatabaseSpy
```

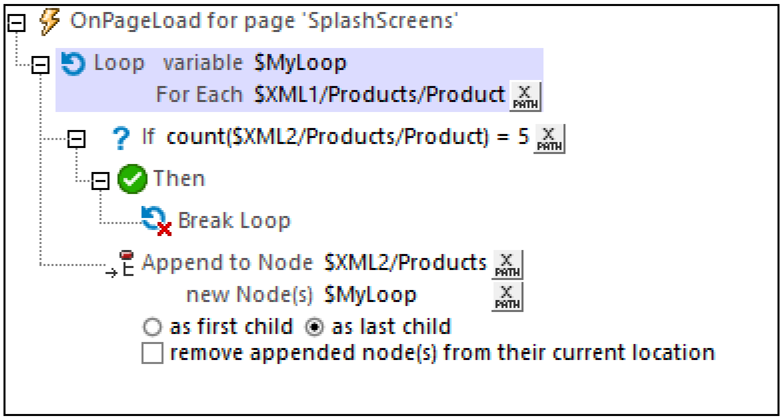
The loop is exited either when all iterations have been completed or a condition is set under which a Break Loop[899] action is executed *(see next topic)*.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

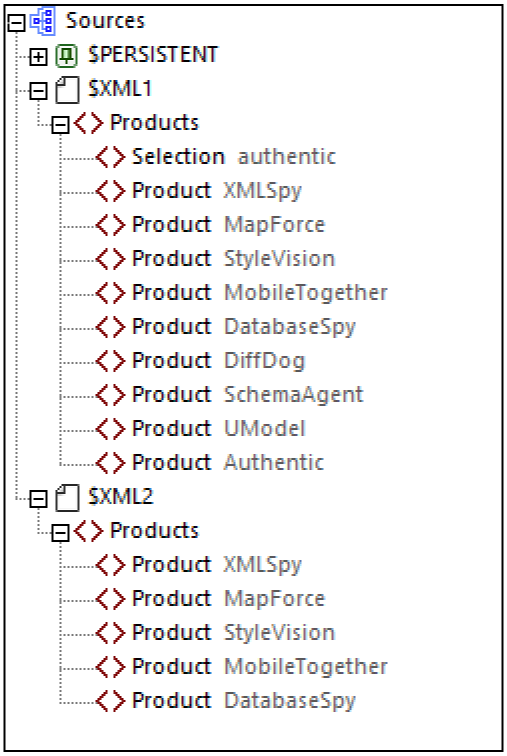
## 10.18.5 Break Loop

The Break Loop action (*see screenshot below*) is used within a loop and is used to exit the loop. If you want the loop to be exited when a specific condition is fulfilled, you must specify this condition in, say, an If-Then[894] action, and place the Break Loop inside the If-Then[894] action *(see screenshot below)*. If you do not place the Break Loop action within a condition, then the loop will be broken directly the Break Loop action is encountered.



```
OnPageLoad for page 'SplashScreens'
  Loop   variable $MyLoop
         For Each $XML1/Products/Product
    If count($XML2/Products/Product) = 5
      Then
        Break Loop
    Append to Node $XML2/Products
         new Node(s) $MyLoop
    ○ as first child ● as last child
    □ remove appended node(s) from their current location
```

The actions that are performed in the loop that is shown in the screenshot above are described in the Loop[897] topic. Here, we focus on the Break Loop action. In our example *(see screenshot above),* the Break Loop action is executed when the number of appended nodes in **`$XML2`** reaches 5. When the iteration over the sixth **`Product`** element of **`$XML1`** starts, the condition condition defined in the **`If`** clause is checked. The XPath expression for the condition is: `count($XML2/Products/Product) = 5`, which now evaluates to **`true()`**. As a result, the **`Then`** clause is executed and the loop is exited. *See screenshot below.*



```
Sources
  $PERSISTENT
  $XML1
    Products
      Selection authentic
      Product XMLSpy
      Product MapForce
      Product StyleVision
      Product MobileTogether
      Product DatabaseSpy
      Product DiffDog
      Product SchemaAgent
      Product UModel
      Product Authentic
  $XML2
    Products
      Product XMLSpy
      Product MapForce
      Product StyleVision
      Product MobileTogether
      Product DatabaseSpy
```

Also see the description of the Loop[897] action *(previous topic)*.

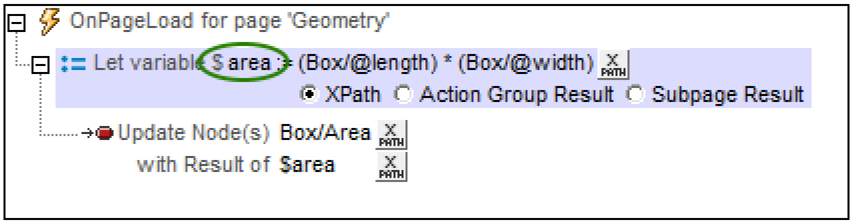
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.18.6 Let

The Let action (*screenshot below*) defines a variable with a value that is set via: (i) an XPath expression, (ii) an Action Group Result, or (iii) a Subpage Result.

- To enter the name of the variable, double-click to the right of the **$** sign and type in the variable's name (*circled in green in screenshot below*).
- Select whether you wish to set the value of the variable via an XPath expression, an Action Group Result, or a Subpage Result.
- Define one or more child actions of the Let action. For example, the Let action shown in the screenshot below contains an Update Node(s)[886] action, which updates a node with the value of the variable defined in the Let action.



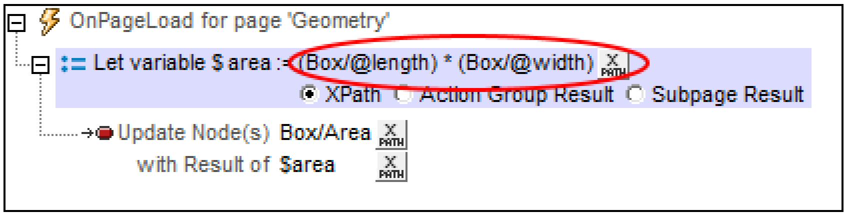
OnPageLoad for page 'Geometry'
Let variable $area := (Box/@length) * (Box/@width)
● XPath ○ Action Group Result ○ Subpage Result
Update Node(s) Box/Area
with Result of $area

**Note:** The variable defined in a Let action is in scope only within that Let action. This means that it can be used only in child actions of the Let action.

**Note:** If a variable contains a nodeset and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. However, this does not apply if only the values of variables in a nodeset have been modified.

## The XPath option

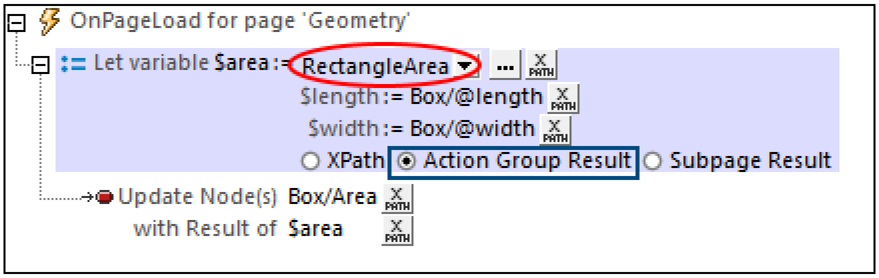
The *XPath* option enables you to enter a static value or to generate a dynamic value using XML tree nodes. For example, in the screenshot below, the values of two nodes are multiplied (*circled in red*). The resulting value will be the value of the variable (`$area`). The variable has then been used to update the content of an XML tree node.



OnPageLoad for page 'Geometry'
Let variable $area := (Box/@length) * (Box/@width)
● XPath ○ Action Group Result ○ Subpage Result
Update Node(s) Box/Area
with Result of $area

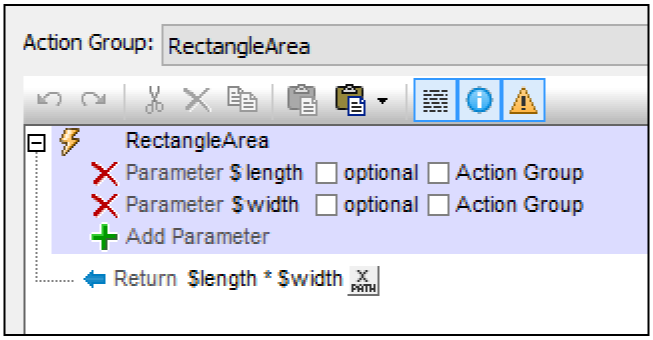
## The Action Group Result option

The *Action Group Result* option (*screenshot below*) sets the value of the variable to be the result of an Action Group. In the screenshot below, we have given the variable a name of `$area`, and set its value to be the result of the Action Group called `RectangleArea` (*circled in red below*). (All existing Action Groups are available for selection in the variable's combo box.) To edit the Action Group, click the **Edit** button (*circled in green*). We have also defined the values of two parameters to be dynamic; they will take their values from XML tree nodes.



OnPageLoad for page 'Geometry'
Let variable $area := RectangleArea
$length := Box/@length
$width := Box/@width
○ XPath ● Action Group Result ○ Subpage Result
Update Node(s) Box/Area
with Result of $area

The Action Group returns a result via the Return[909] action (*see screenshot below*). In the screenshot below, for example, we declare two parameters (`$length` and `$width`); in the Return[909] action, we multiply the values of the two parameters. Note that the values of the parameters are obtained at runtime from the XML tree nodes that have been defined, in the Let action (*see screenshot above*), as the values of the Let action's parameters.

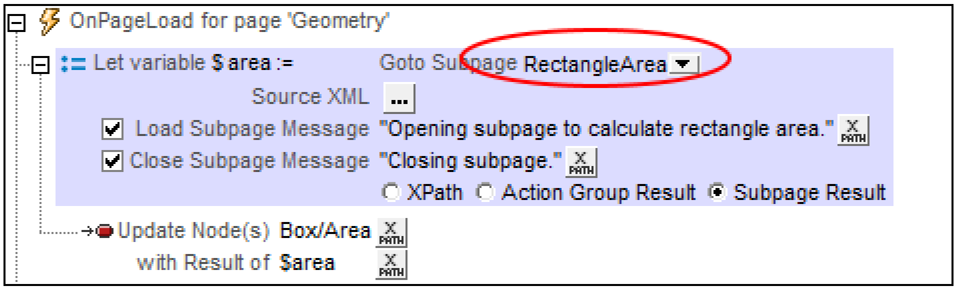

Action Group: RectangleArea

RectangleArea
Parameter $length □ optional □ Action Group
Parameter $width □ optional □ Action Group
Add Parameter
Return $length * $width

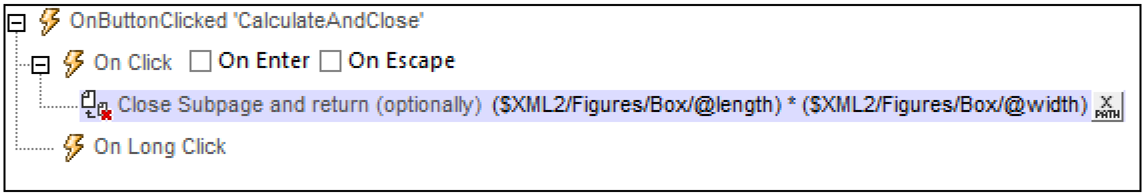
## The Subpage Result option

The *Subpage Result* option (*screenshot below*) sets the value of the variable to be the result of a subpage. This allows a calculation to be carried out on another page. When the subpage is closed, a result can be optionally returned. This result is the Subpage Result that will be used as the value of the variable defined in the Let action. For example, in the screenshot below, the Let action defines a variable called `$area`, and sets its value to be the result of the subpage called `RectangleArea` (*circled in red below*).

The properties of the *Subpage Result* option are the same as the Go to Subpage[783] action, and are described in detail there.


OnPageLoad for page 'Geometry'
Let variable $area := Goto Subpage RectangleArea
Source XML ...
☑ Load Subpage Message "Opening subpage to calculate rectangle area."
☑ Close Subpage Message "Closing subpage."
○ XPath ○ Action Group Result ● Subpage Result
Update Node(s) Box/Area
with Result of $area

At runtime, when the Let action is executed, the subpage is opened and it will be processed as defined in its design. The subpage will be closed when the Close Subpage[788] action is executed. This action has an optional return value which is calculated by an XPath expression (*see screenshot below*). This returned value will be passed to the Let action and will become the value of the variable defined in the Let action.


OnButtonClicked 'CalculateAndClose'
On Click □ On Enter □ On Escape
Close Subpage and return (optionally) ($XML2/Figures/Box/@length) * ($XML2/Figures/Box/@width)
On Long Click

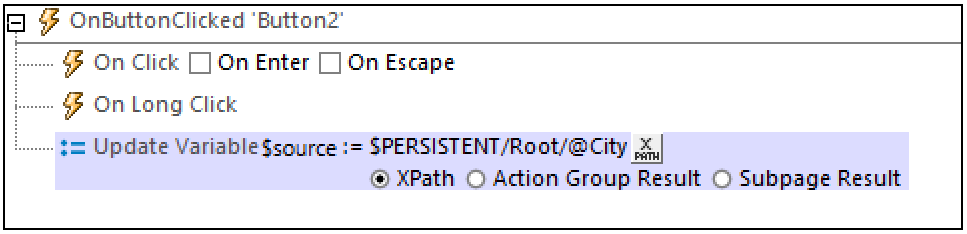
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.18.7 Update Variable

The action *(screenshot below)* updates the value of an existing user-defined variable[1309] that you identify by selecting it in the action (`$area` in the screenshot below). The new value to update with will be one of the following:

- the result of evaluating an XPath expression that you enter
- an Action Group result
- a subpage result



```
OnButtonClicked 'Button2'
    On Click □ On Enter □ On Escape
    On Long Click
    Update Variable $source := $PERSISTENT/Root/@City
        ● XPath ○ Action Group Result ○ Subpage Result
```

The variable to be updated can be any user-defined variable[1309] that is in scope at the point when the Update Variable action is triggered. These variables include: user-defined global variables[1309]; variables defined in the Let[900] and Try/Catch Exceptions[907] actions; variables in Action Groups[942]; the parameters of a sub page[381]; and the parameters and variables of control templates[1201]. (Note that the Update Variable action cannot be applied to Loop variables[897].)

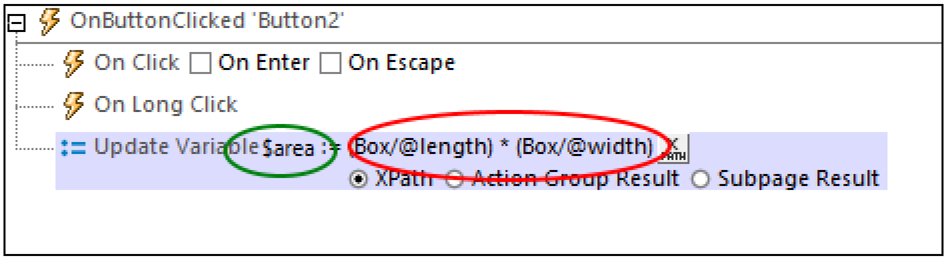
Double-click the field (circled green), and either enter the name of your user-defined variable[1309] or select it from the list that appears.

At run time, if no variable with the given name is found, then an error message to this effect is generated together with a list of variables that are in scope at that point and which could potentially be updated.

**Note:** Application-defined variables (such as Dynamic Local Variables[1304] and Static Global Variables)[1300] **cannot** be modified by the Update Variable action.

### The XPath option

The *XPath* option enables you to enter a static value or to generate a dynamic value using XML tree nodes. For example, in the screenshot below, the values of two nodes are multiplied (*circled in red*). The resulting value will be passed to the user-defined variable **`$area`**.



⊟ OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

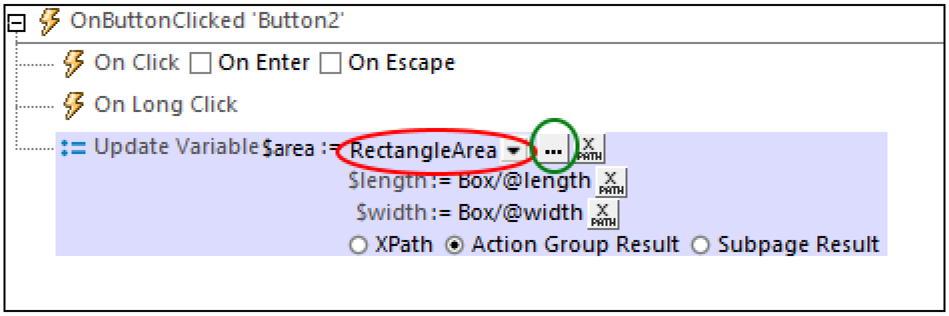
On Long Click

Update Variable $area := (Box/@length) * (Box/@width)

● XPath ○ Action Group Result ○ Subpage Result

## The Action Group Result option

The *Action Group Result* option (*screenshot below*) sets the value of the variable to be the result of an Action Group. In the screenshot below, we have given the variable a name of **`$area`**, and set its value to be the result of the Action Group called **`RectangleArea`** (*circled in red below*). (All existing Action Groups are available for selection in the variable's combo box.) To edit the Action Group, click the **Edit** button (*circled in green*). We have also defined the values of two parameters to be dynamic; they will take their values from XML tree nodes.



⊟ OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

On Long Click

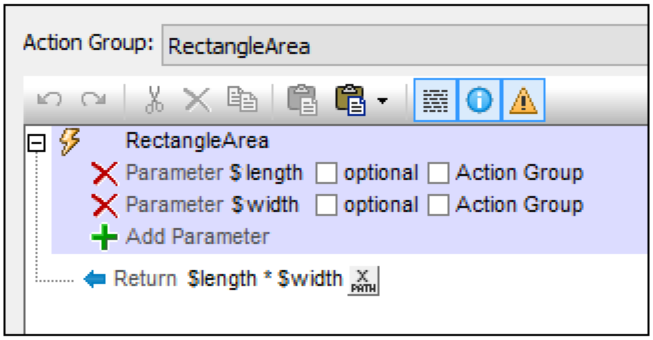
Update Variable $area := RectangleArea ▾ ...

$length:= Box/@length

$width:= Box/@width

○ XPath ● Action Group Result ○ Subpage Result

The Action Group returns a result via the Return[909] action (*see screenshot below*). In the screenshot below, for example, we declare two parameters (`$length` and `$width`); in the Return[909] action, we multiply the values of the two parameters. Note that the values of the parameters are obtained at runtime from the XML tree nodes that have been defined, in the Let action (*see screenshot above*), as the values of the Let action's parameters.



Action Group: RectangleArea

⊟ RectangleArea

Parameter $length □ optional □ Action Group

Parameter $width □ optional □ Action Group

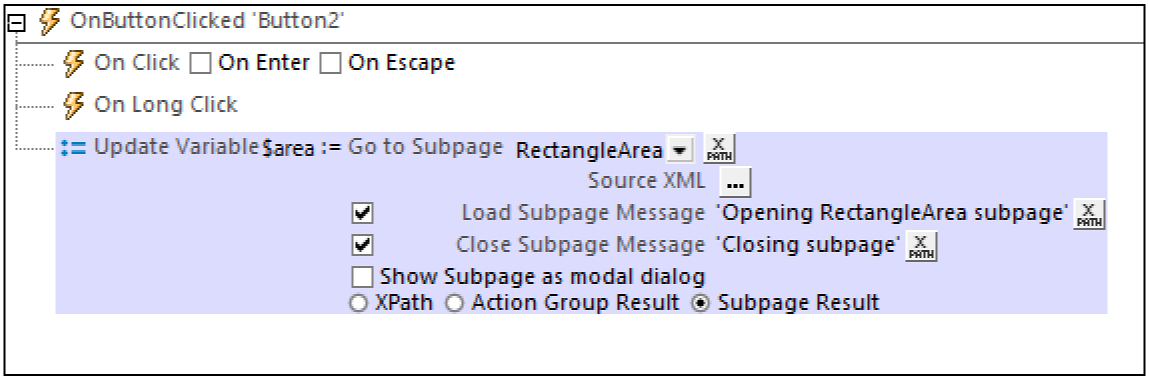
Add Parameter

Return $length * $width

## The Subpage Result option

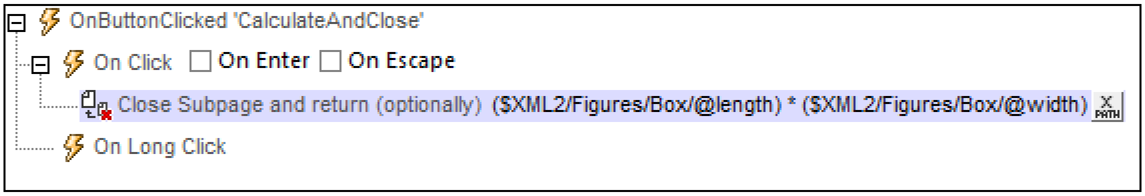
The *Subpage Result* option (*screenshot below*) sets the value of the variable to be the result of a subpage. This allows a calculation to be carried out on another page. When a subpage is closed, it can optionally return a result. This result is the Subpage Result that will be used as the value of the variable defined in the Update Variable action. For example, in the screenshot below, the Update Variable action defines a variable called **`$area`** and sets its value to be the result of the subpage called **`RectangleArea`**.

The properties of the *Subpage Result* option are the same as the Go to Subpage[783] action, and are described in detail there.



OnButtonClicked 'Button2'
On Click □ On Enter □ On Escape
On Long Click
Update Variable $area := Go to Subpage RectangleArea
Source XML
☑ Load Subpage Message 'Opening RectangleArea subpage'
☑ Close Subpage Message 'Closing subpage'
□ Show Subpage as modal dialog
○ XPath ○ Action Group Result ● Subpage Result

At runtime, when the Let action is executed, the subpage is opened and it will be processed as defined in its design. The subpage will be closed when the Close Subpage[788] action is executed. This action has an optional return value which is calculated by an XPath expression (*see screenshot below*). This returned value will be passed to the Let action and will become the value of the variable defined in the Let action.



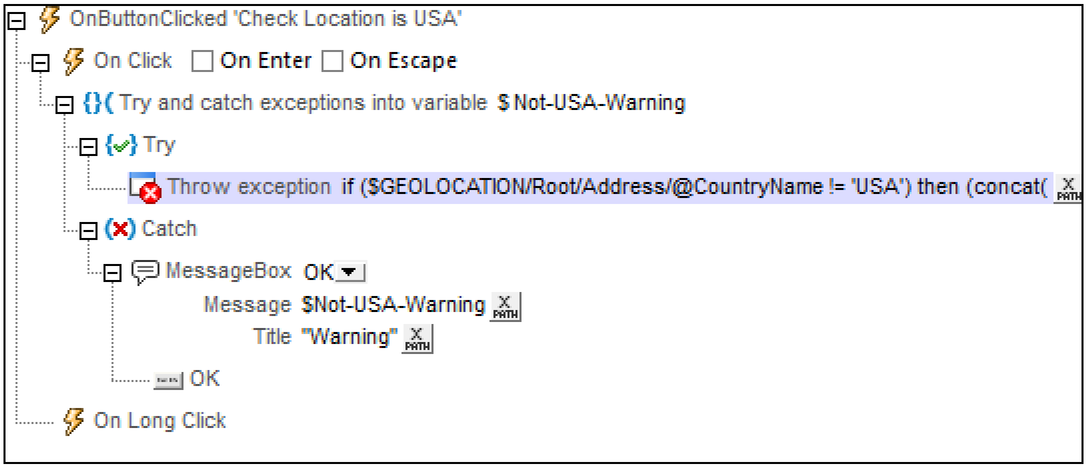
OnButtonClicked 'CalculateAndClose'
On Click □ On Enter □ On Escape
Close Subpage and return (optionally) ($XML2/Figures/Box/@length) * ($XML2/Figures/Box/@width)
On Long Click

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.18.8 Throw

The Throw action is intended to be used in the Try part of a Try/Catch action[905] (*see screenshot below*). It evaluates an XPath expression. If the result of the evaluation is not an empty sequence, then an exception is thrown, and the exception is stored in the variable of the Try/Catch action[905]; *in the screenshot below, this variable is named* `$Not-USA-Warning`.



OnButtonClicked 'Check Location is USA'
On Click □ On Enter □ On Escape
{}( Try and catch exceptions into variable $Not-USA-Warning
{✓} Try
Throw exception if ($GEOLOCATION/Root/Address/@CountryName != 'USA') then (concat(
(×) Catch
MessageBox OK
Message $Not-USA-Warning
Title "Warning"
OK
On Long Click

In the example shown in the screenshot above, we throw an exception if the geolocation of the device[733] is not in the USA. The XPath expression is:

```
if ($MT_GEOLOCATION/Root/Address/@CountryName != 'USA')
then (concat( 'Warning: Device location is outside the US: ',
$MT_GEOLOCATION/Root/Address/@CountryName))
else ()
```

This expression works as follows:

- The `if` clause checks whether the value of the `$MT_GEOLOCATION/Root/Address/@CountryName` node is (not) `'USA'`.
- The `then` clause is processed if the country name **is not** `USA`. This clause generates a string.
- The `else` clause is processed if the country name **is** `USA`. It produces an empty sequence

If the geolocation country **is not** `USA`, then the condition is `true` and the expression evaluates to the string generated by the `then` clause. Since this result is not an empty sequence, an exception is thrown and the generated string is stored in the Try/Catch[905] variable `$Not-USA-Warning`.

If the geolocation country **is** `USA`, then the condition is `false` and the expression evaluates to an empty sequence (generated by the `else` clause). Because the result is an empty sequence, no exception is thrown. Therefore, the Catch part of the Try/Catch action[905] is not executed.

**Note:** If a sequence contains an empty string item **`('')`**, then the sequence is **not** empty (and an exception will be thrown).

The tutorial Sharing Geolocations[230] shows how the Try/Catch[905] and Throw[905] actions can be used.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.18.9 Try/Catch Exceptions

The Try/Catch Exceptions action has two parts (*highlighted in the screenshot below*):

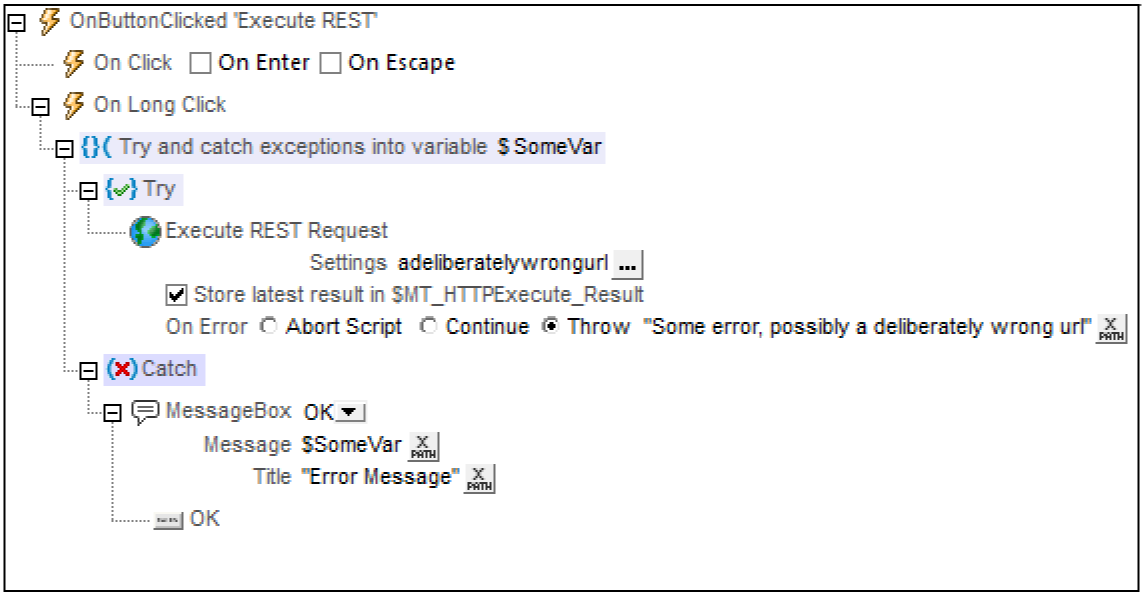
- *Try:* Defines a condition or an action to try.
  - A condition is defined in the XPath expression of a Throw action[905]. (*See the Sharing Geolocations tutorial[233] for an example of using the Throw action[905].*)
  - If an action is defined (such as the Execute REST Request action in the screenshot below) and an error is detected while executing the action, then you can choose from among the following **options**: (i) abort the action; (ii) ignore the error and continue; or (iii) throw an exception that is stored in the Try/Catch action's variable; this is the *Throw* **option**. (Even if you choose to continue (the second option), you can still throw an error by using the Throw action[905].)
  - Both the Throw action[905] (defined for a condition) or the Throw option (defined for an action) each throw an exception that is stored in the Try/Catch action's variable.
- *Catch:* Defines actions to execute if, and only if, an exception is thrown (*see the description of the screenshot below)*. If no Catch action is defined, then the action that follows the Try/Catch action is processed.

**Note:** Exceptions can be thrown in two ways: via a Throw action[905] (defined for a condition) and via a Throw option (defined for an action).

**Note:** If an exception is thrown, it is stored in the variable of the Try/Catch action, and the Catch part will be executed.

**Note:** If no exception is thrown in the Try part of the action (by the Throw action/option), then the Catch part is **not** executed.

**Note:** If a variable contains a nodeset and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. However, this does not apply if only the values of variables in a nodeset have been modified.



OnButtonClicked 'Execute REST'
On Click □ On Enter □ On Escape
On Long Click
{}( Try and catch exceptions into variable $SomeVar
{✓} Try
Execute REST Request
Settings adeliberatelywrongurl ...
☑ Store latest result in $MT_HTTPExecute_Result
On Error ○ Abort Script ○ Continue ● Throw "Some error, possibly a deliberately wrong url"
(✗) Catch
MessageBox OK
Message $SomeVar
Title "Error Message"
OK

In the Try/Catch action shown in the screenshot above, we have done the following:

1. We have given the Try/Catch variable a name of `$SomeVar` (by double-clicking to the right of the **$** symbol, and entering the name).
2. In the Try part of the action, we have set up the Execute REST request 837.
3. In the Try part of the action, we have selected the *Throw* option for the *On Error* sub-action of the Execute REST request 837, and entered an exception message as the option's XPath expression. As a result, if an error is detected, an exception is thrown and the exception message is stored in the `$SomeVar` variable.
4. In the Catch part of the action, we have defined a Message Box 679 action to show the message that is stored in the `$SomeVar` variable.

**Note:** Besides the Throw **option** described above, a Throw **action** is also available. Instead of using the *Throw* option of the *On Error* sub-action (*as described in Step 2 above*), you could use the *Continue* option and insert a Throw 905 action in the *Continue* option's *On Error* sub-action.

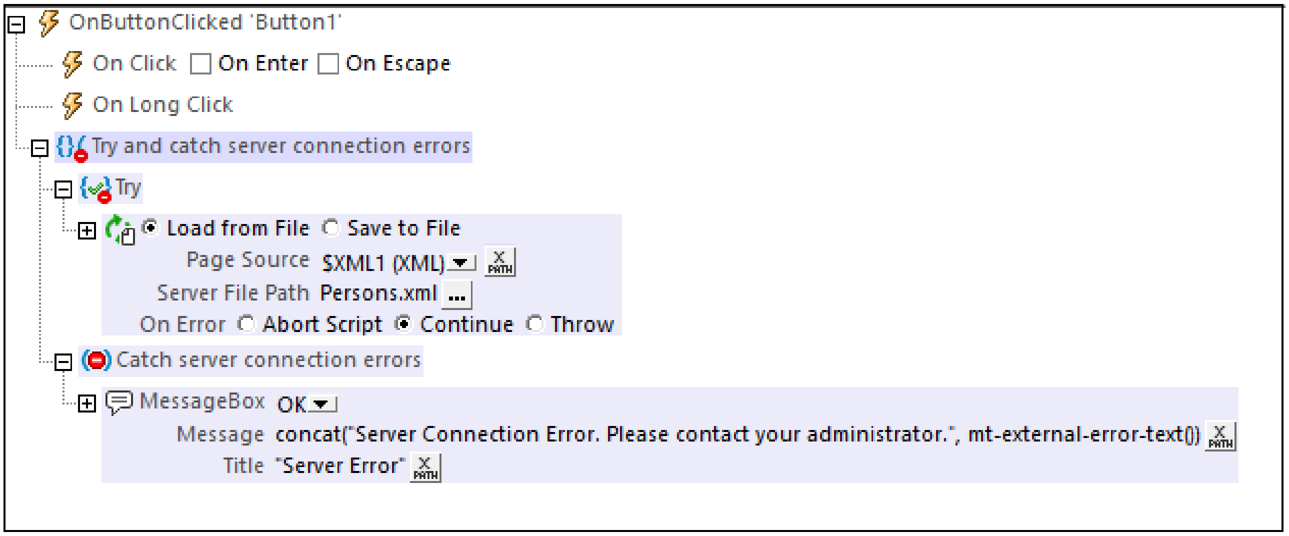
The tutorial Sharing Geolocations 230 shows how the Try/Catch 905 and Throw 905 actions can be used.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.18.10 Try/Catch Server Connection

The *Try* part of this action (*see screenshot below*) is used to try actions that will make a connection to the server. If the connection is not made, then the *Catch* part of the action is triggered. If the connection can be made, but there might be subsequent exceptions (for example if a file is not found), then you can use the *Throw* option to generate the exceptions (*see screenshot below*).



OnButtonClicked 'Button1'

- On Click □ On Enter □ On Escape
- On Long Click
- Try and catch server connection errors
  - Try
    - ● Load from File ○ Save to File
      - Page Source $XML1 (XML)
      - Server File Path Persons.xml
      - On Error ○ Abort Script ● Continue ○ Throw
  - Catch server connection errors
    - MessageBox OK
      - Message concat("Server Connection Error. Please contact your administrator.", mt-external-error-text())
      - Title "Server Error"

**Note:** The MobileTogether XPath extension function `mt-external-error-text()` can be used in the *Catch* part to display information about the server connection error.

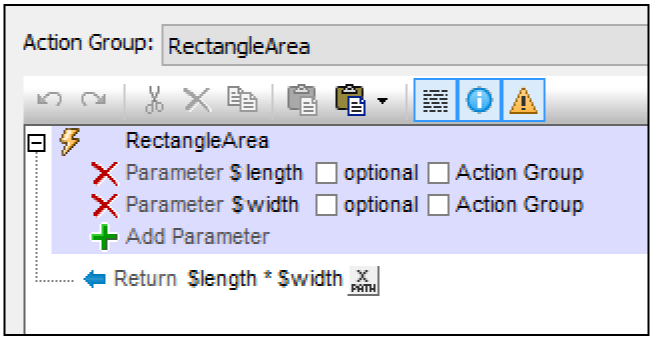
**Note:** When a server connection error occurs, the first of the following actions that exists is triggered: (i) a Try/Catch Server Connection Errors (908) action, (ii) action/s for the **`OnServerConnectionError`** (394) event, (iii) a MobileTogether message about the error, following which the workflow is resumed.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions (1262) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box (679) action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions (1262).

## 10.18.11 Return

In a Return action, an XPath expression defines the value that is returned by an Action Group. This value is called the Action Group Result, and it can be used to set the value of a variable defined in a Let action (900). In the screenshot below, for example, the Return action defines the value that is returned by the Action Group named `RectangleArea`. The Return action's XPath expression can contain parameters declared in the Action Group. The values of the parameters are supplied at runtime by the Let action (900) that calls the Action Group. See the section *Setting Variable Values via Action Groups* (950) for a description and example of how the Return action can be used.



Action Group: RectangleArea

RectangleArea
Parameter $length □ optional □ Action Group
Parameter $width □ optional □ Action Group
Add Parameter
Return $length * $width

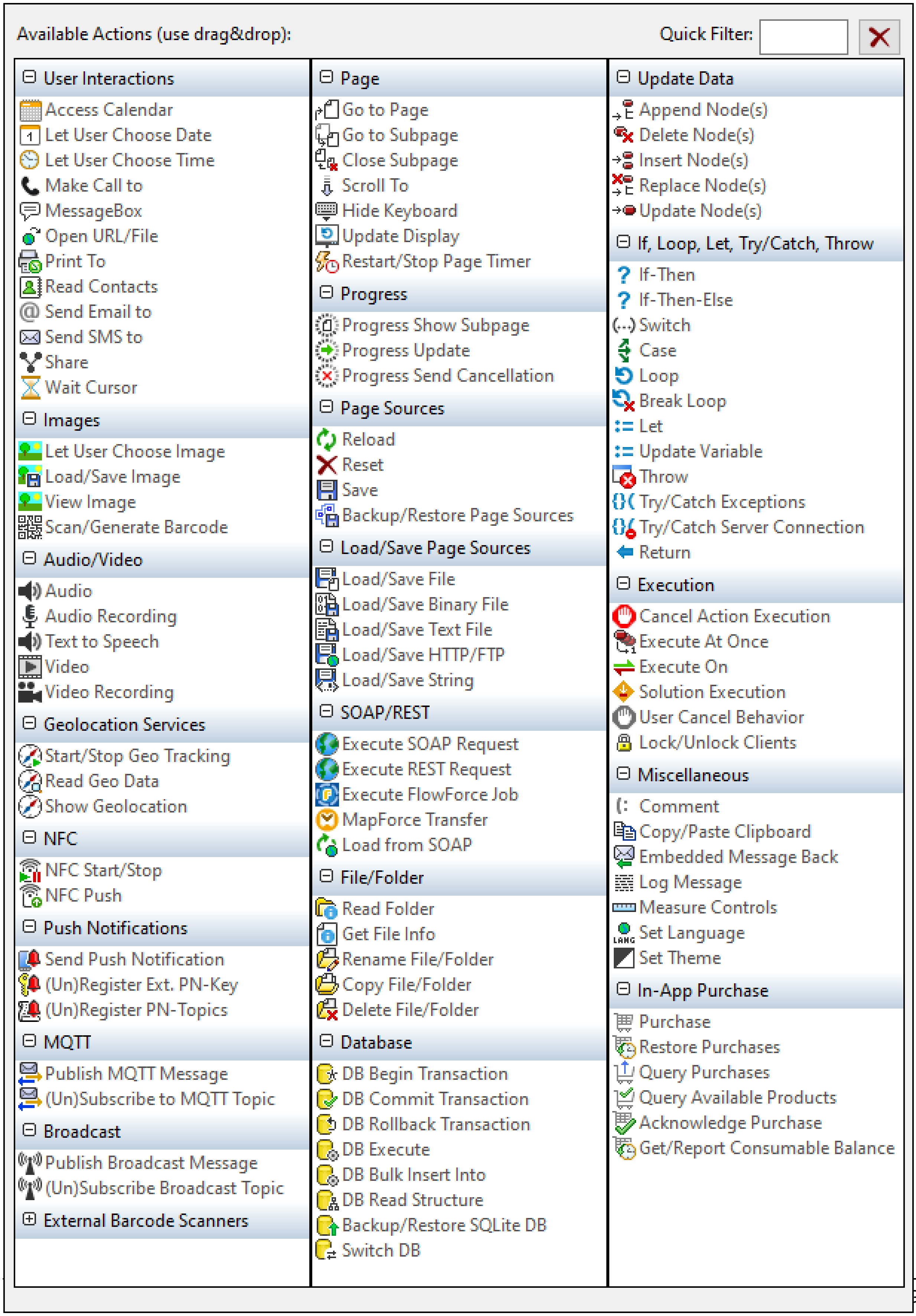
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

# 10.19 Execution

The following actions are available in the Execution group of the Actions dialog (*screenshot below*):

- Cancel Action Execution 913
- Execute at Once 914
- Execute On 914
- Solution Execution 915
- User Cancel Behavior 917
- Lock/Unlock Clients



Available Actions (use drag&drop):

Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

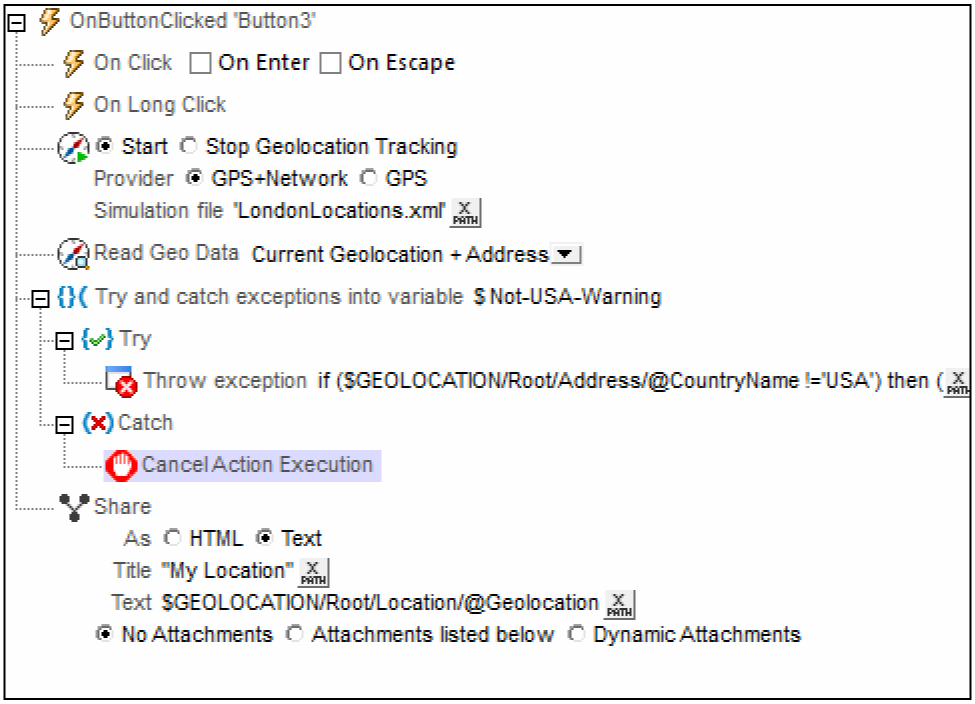
⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events*[389] *and* *Control Events*[665].

## 10.19.1 Cancel Action Execution

The Cancel Action Execution action (*highlighted in the screenshot below*) cancels the execution of the event's action sequence. In the screenshot below, for example, if an exception is thrown in Try part of the Try/Catch action[907], then the Catch part of the Try/Catch action[907] is executed. Since this part contains the Cancel Action Execution action, the event's sequence of actions is canceled. As a result, the Share action[699] will not be executed.



```
OnButtonClicked 'Button3'
    On Click  □ On Enter  □ On Escape
    On Long Click
    ● Start  ○ Stop Geolocation Tracking
    Provider  ● GPS+Network  ○ GPS
    Simulation file  'LondonLocations.xml'
    Read Geo Data  Current Geolocation + Address
    {}( Try and catch exceptions into variable  $Not-USA-Warning
        Try
            Throw exception  if ($GEOLOCATION/Root/Address/@CountryName !='USA') then (
        Catch
            Cancel Action Execution
    Share
        As  ○ HTML  ● Text
        Title  "My Location"
        Text  $GEOLOCATION/Root/Location/@Geolocation
        ● No Attachments  ○ Attachments listed below  ○ Dynamic Attachments
```

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.19.2 Execute at Once

If a variable contains a nodeset from a page source and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. There are two situations in which alternative processing is desirable:

- Changes to a nodeset occur via a sequence of multiple actions, each of which changes the nodeset. This leads to a situation in which each action is processing a nodeset that has been modified by a previous action.
- The evaluation of an XPath expression containing a variable that references a modified page source (for example, if the XPath expression occurs within a Loop action).

The **Execute at Once** action does the following:

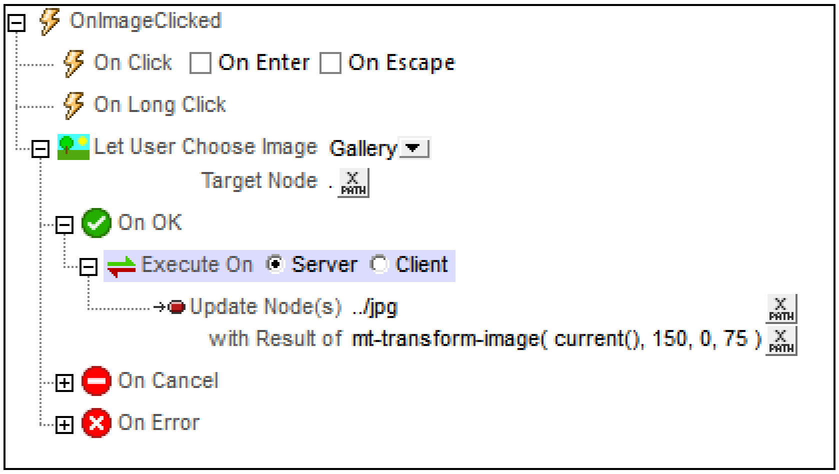
1. Evaluates the XPath/XQuery statements of all its child actions
2. Applies the effects of these evaluations to the relevant nodesets of the page.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.19.3 Execute On

The Execute On action (*see screenshot below*) explicitly specifies where the action's sub-actions must be executed: on the server or the client.



OnImageClicked
On Click □ On Enter □ On Escape
On Long Click
Let User Choose Image Gallery
Target Node .
On OK
Execute On ● Server ○ Client
Update Node(s) ../jpg
with Result of mt-transform-image( current(), 150, 0, 75 )
On Cancel
On Error

The screenshot above displays how the Execute On action is typically used:

1. A Let User Choose Image[706] action prompts the user to select an image from a gallery and save the image as Base64 to the current node (which is located, say, with `//image/base64`).
2. If the image is successfully transferred to the current node, the *On OK* condition uses the Execute On action to transform the user-selected image on the server (with the Altova XPath extension function **`mt-transform-image`**[1718]) and then update the sibling **`jpg`** node. The node is updated on the server, and, when the processing of actions is completed, is transferred to the client.

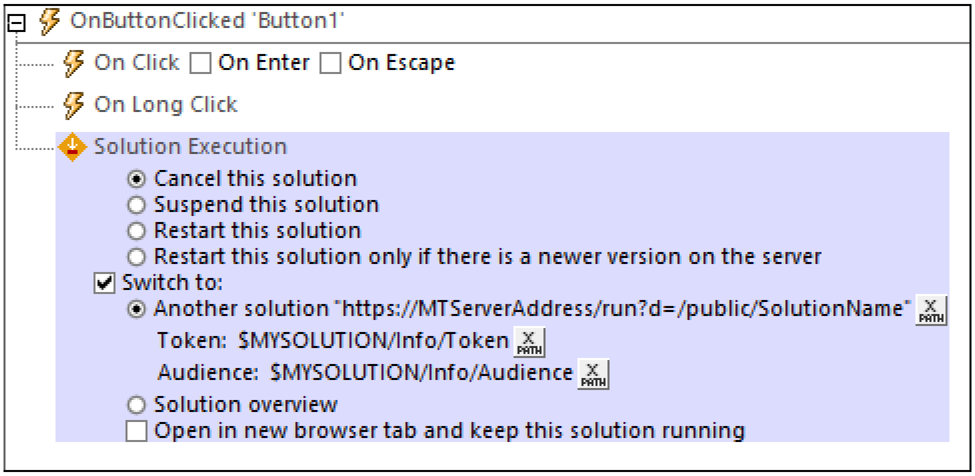
> *Transformation on client or server*
> The function **`mt-transform-image`**[1718] will be executed on the client if not explicitly specified otherwise. This could create memory problems on some clients. When the transformation is started, the image is unpacked from the format of its Base64 encoding to a BMP format, which could be very large. After the transformation is completed, the transformed file is stored back to the original format. The large BMP format could create memory problems on some clients, and you should be aware of this.
>
> To avoid the possibility of memory problems on the client, explicitly specify that the transformation must be carried out on the server. Do this with the Execute On action[914], specifying that the child actions be performed on the server. All the child actions of this Execute On action[914] will then be carried out on the server. You can use an action such as Update Node[886] to update a node with the result of a transformation. The target node will be updated with the transformed image. MobileTogether automatically transfers the results to the client when action handling is complete or when the workflow switches back to the client.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.19.4 Solution Execution

When the event is triggered, the Solution Execution action provides you with the option to: (i) cancel the solution, (ii) leave it suspended (that is, running in the background), (iii) restart the solution (with the same parameters as on the solution start), or (iv) restart the solution only if there is a newer version of the solution on the server..



⊟ OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

Solution Execution

● Cancel this solution
○ Suspend this solution
○ Restart this solution
○ Restart this solution only if there is a newer version on the server

☑ Switch to:

● Another solution "https://MTServerAddress/run?d=/public/SolutionName"

Token: $MYSOLUTION/Info/Token

Audience: $MYSOLUTION/Info/Audience

○ Solution overview

□ Open in new browser tab and keep this solution running

If you select either of the first two options (cancel or suspend), you can do one of the following: (i) switch to another solution, (ii) switch to the Solutions Overview (the *Solutions* page of MobileTogether Client), (iii) do not switch to either another solution or the Solutions Overview. The various possibilities are given in the table below.

| Switch to... | Cancel solution | Suspend solution |
|---|---|---|
| **Another solution** | Solution is closed without displaying any message, and the specified solution is opened. | The current solution runs "minimized" in the background, and the specified solution is opened. |
| **Solution Overview** | Solution is closed without displaying any message, and the display switches to the *Solutions Overview* page of MobileTogether Client. | The current solution runs "minimized" in the background, and the display switches to the *Solutions Overview* page of MobileTogether Client."Switch to" |
| **"Switch to" unchecked** | Solution is closed without displaying any message, and the display switches to where it was before the solution was started (*see notes below*). | The current solution runs "minimized" in the background, and the display switches to where it was before the solution was started (*see notes below*). |

Note the following points:

- Web clients do not support suspended solutions. However, the current solution can be kept running while another solution is opened in another browser tab and becomes the active solution.
- The *Open in new browser tab* option is available for web clients only. It starts the solution in a new browser tab; the original solution is not stopped.
- A solution that is suspended (running in the background) is displayed minimized as an icon in the *Running* page of the MobileTogether Client application, and can be opened by tapping this icon.
- If the solution runs "minimized" in the background, then the solution is paused at that point, and no further solution action is executed. For example, no timers are executed, no geolocations are used, and audio playback is paused. When the solution is re-opened, actions defined for the *On Reopen* option of the `OnPageRefresh` 390 event are executed, and audio playback that was paused is resumed. *Also see the project property On Switch to Other Solution* 296.

- The solution to switch to is specified by clicking the Edit XPath Expression[1244] button and entering the location of the solution as a string (that is, within quotes). The location must be on the same server as the current solution and the location string must be exactly the same as the string that was entered when the target solution was deployed[1574]. *See screenshot above*. If the target solution is already running (minimized), then it is opened and continues from where it was when it was switched to its minimized state.
- For Web clients only: When the option to switch to a new solution is selected, the *Token and Audience*[1237] settings can be entered as XPath string expressions. These two entries are required only for setting up solutions for authenticated users[1237]; they can be left empty if you do not want to silently authenticate users when they run the second solution.
- If the *Switch to* option is unchecked, the display either: (i) returns to the home screen if the solution was started via its shortcut, or (ii) returns to the *Solutions Overview*. On iOS, it always returns to the *Solutions Overview* page.
- Tapping the **Back** button on the top page of a solution running in parallel suspends the solution. (i) *On Android, Windows App and Windows*: It returns to the home screen if the solution was started via its shortcut; otherwise it returns to the *Solutions Overview*. In either case, the return action is performed directly, without any end-user input. (ii) *On iOS (and in non-parallel solutions)*: The user is asked whether the solution should be exited or not. If the response is to exit, then the user is returned to the *Solutions Overview* page.

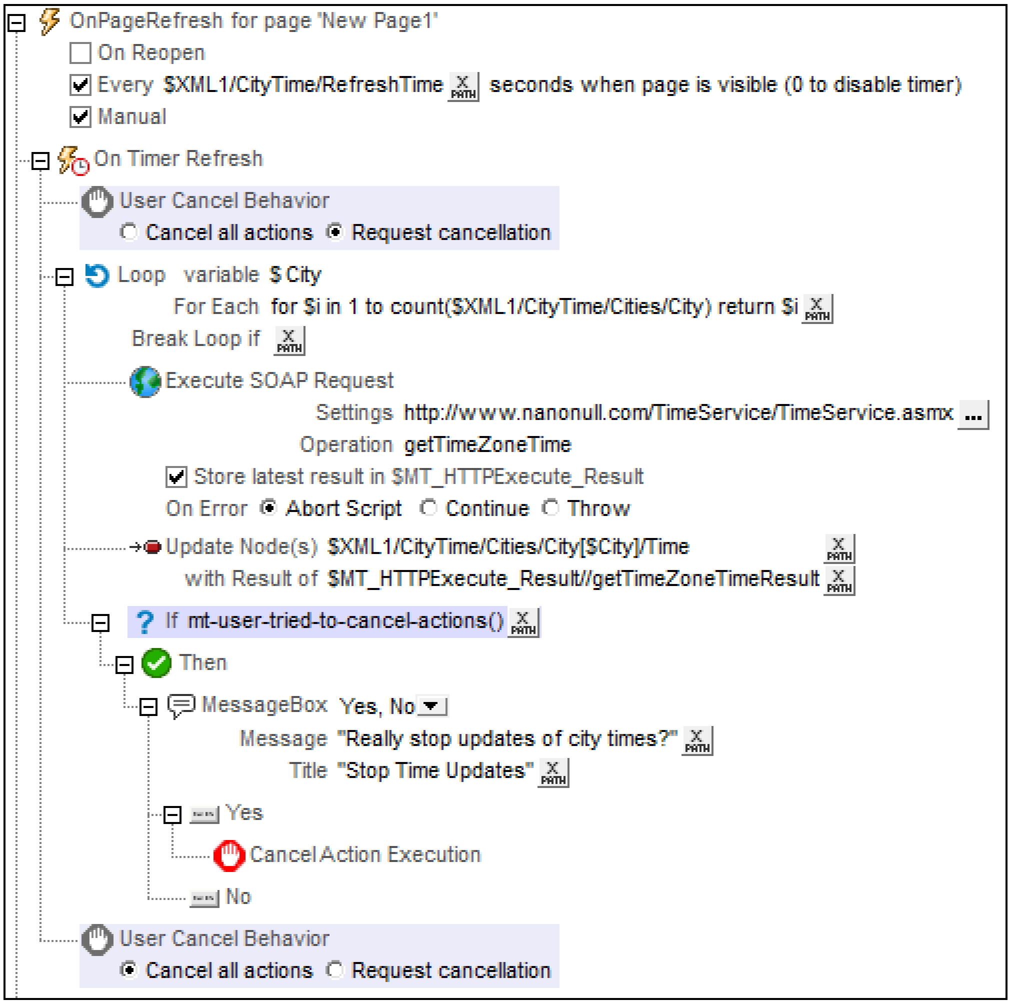
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.19.5 User Cancel Behavior

The User Cancel Behavior action (*highlighted in the screenshot below*) enables you to override a user cancellation action and continue with action execution. The following options are available:

- *Cancel all actions:* If the user presses the **Back** button (or the **Cancel** button that appears during the execution of long actions), the cancellation is allowed. This is the default behavior.
- *Request cancellation:* This option enables you to override a user cancellation action. *See the example in the screenshot below.* If the user attempts to cancel action execution, the action execution is not interrupted. Instead, the **`mt-user-tried-to-cancel-actions`**[1262] function is set to **`true`**. You can then define a set of appropriate action/s to take depending on the value of this function. After the action's *Request cancellation* flag has been set to true, the flag can be reset by using the User Cancel Behavior action again, this time with the option set to *Cancel all actions*. This resets the **`mt-user-tried-to-cancel-actions`**[1262] function to its default value of **`false`**.



```
⊟ OnPageRefresh for page 'New Page1'
    □ On Reopen
    ☑ Every $XML1/CityTime/RefreshTime seconds when page is visible (0 to disable timer)
    ☑ Manual
  ⊟ On Timer Refresh
      User Cancel Behavior
        ○ Cancel all actions ● Request cancellation
    ⊟ Loop  variable $City
            For Each  for $i in 1 to count($XML1/CityTime/Cities/City) return $i
        Break Loop if
        Execute SOAP Request
                    Settings  http://www.nanonull.com/TimeService/TimeService.asmx ...
                   Operation  getTimeZoneTime
          ☑ Store latest result in $MT_HTTPExecute_Result
          On Error ● Abort Script ○ Continue ○ Throw
        Update Node(s)  $XML1/CityTime/Cities/City[$City]/Time
          with Result of  $MT_HTTPExecute_Result//getTimeZoneTimeResult
      ⊟ If mt-user-tried-to-cancel-actions()
        ⊟ Then
          ⊟ MessageBox  Yes, No
                Message "Really stop updates of city times?"
                  Title "Stop Time Updates"
            ⊟ Yes
                Cancel Action Execution
              No
      User Cancel Behavior
        ● Cancel all actions ○ Request cancellation
```

In the example shown in the screenshot above, the following is defined for the *On Timer Refresh* [390] event option:

1. The User Cancel Behavior action is set to *Request cancellation*. Consequently, if the user presses either the **Back** or **Cancel** button, the `mt-user-tried-to-cancel-actions` [1262] function will be set to `true`.
2. A Loop action [897] is started. With each iteration, successive `//City` elements are updated, via SOAP requests, with the current time of that city. If there are several `//City` elements and the updates are taking too long, then the end user might try to cancel the updates by pressing either the **Back** or **Cancel** button.
3. At the end of the iteration during which the end user tries to cancel, the `mt-user-tried-to-cancel-actions` [1262] function is evaluated. Since its value at this point will be `true` (*see Point 1 above*), a message box is displayed that asks the user whether to go ahead with a cancellation or not. According to the response, the action is either canceled or continued.
4. After the loop is completed, the User Cancel Behavior action is explicitly set to its default value of *Cancel all actions.* This step is required if you want to set the user-cancellation behavior back to its default (of canceling without requiring confirmation). Otherwise the User Cancel Behavior action will continue to have a value of *Request cancellation* **for all subsequent actions—**till it is explicitly changed.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

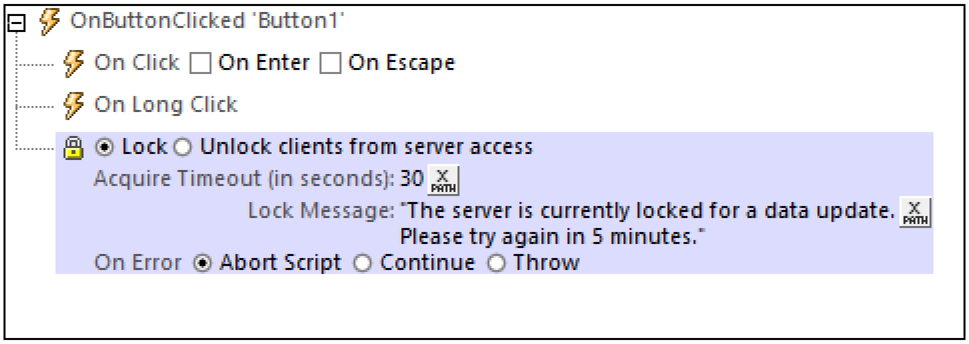
## 10.19.6 Lock/Unlock Clients

The Lock/Unlock Clients action is useful if you want to lock the clients of a solution from accessing the server. This might be the case, for example, if you want to update a database on the server with new data. In such cases, you can lock off the server for all clients of the solution, carry out the server-side actions that the locking client initiates without interruption from other clients, and then (when done) unlock the server for all clients of the solution.

**Note:** This action can be deployed to **MobileTogether Server Advanced Edition only**, not to the standard edition of MobileTogether Server. :

*Lock Clients*

The Lock Clients action (*screenshot below*) locks clients of the current solution from server access.



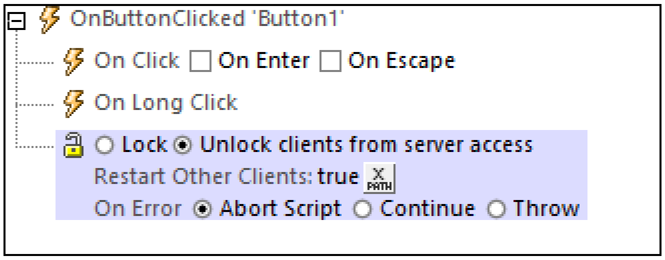
OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
● Lock ○ Unlock clients from server access
Acquire Timeout (in seconds): 30
Lock Message: "The server is currently locked for a data update. Please try again in 5 minutes."
On Error ● Abort Script ○ Continue ○ Throw

The following settings are available:

- *Acquire Timeout:* Specifies the maximum amount of time in seconds before the server is locked for clients of the current solution. If no client of the solution is currently accessing the server, then the server is locked immediately. If clients are currently accessing the solution, then a lock is attempted when the timeout period expires. If the server cannot be locked, an error will be returned. You can set up actions to deal with such an error (*see* Error processing *below*).
- *Lock Message:* This is the message that will be displayed to clients that try to connect to the server while it is locked.

*Unlock Clients*

The Unlock Clients action (*screenshot below*) enables clients that have been locked from server access to access the server again. You can specify whether other clients must be restarted or not (the default value for this setting is **`true`**). Restarting other clients of the solution would enable them to receive the latest changes on the server.



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

○ Lock ● Unlock clients from server access

Restart Other Clients: true

On Error ● Abort Script ○ Continue ○ Throw

Typically, you should set an Unlock Clients action as the final action in the set of server actions that are carried out after locking the server. However, even if you do not set an Unlock Clients action, the server will still be unlocked for clients after all the server actions have been carried out. In this case, an error will be reported and all clients are restarted.

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable [907]. The Catch part of the Try/Catch action [907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action [907] for details.

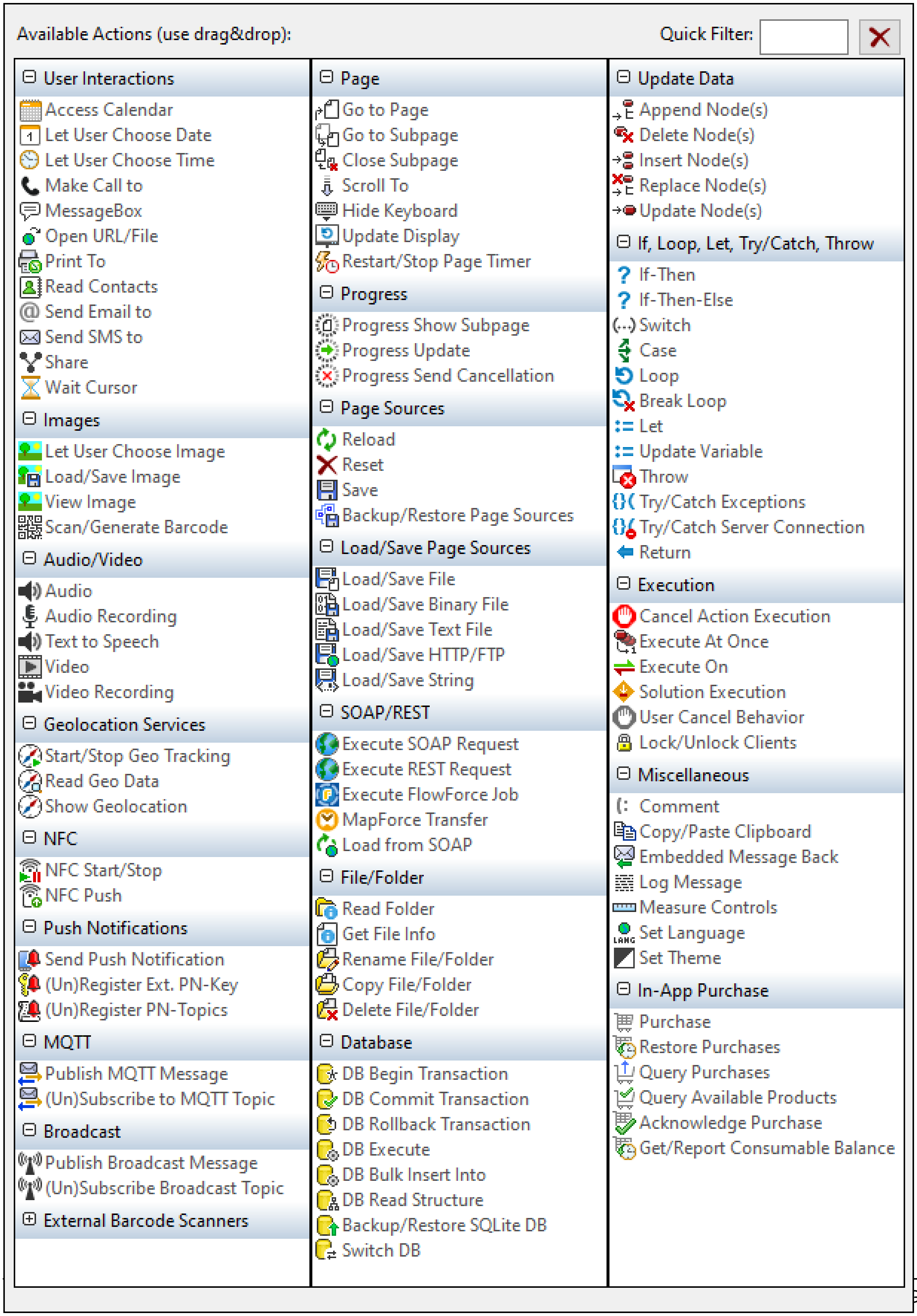
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions [1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box [679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions [1262].

## 10.20 Miscellaneous

The following actions are available in the Miscellaneous group of the Actions dialog (*screenshot below*):

- Comment 923
- Copy/Paste Clipboard 923
- Embedded Message Back 924
- Log Message 925
- Measure Controls 927
- Set Language 928
- Set Theme



Available Actions (use drag&drop):

Quick Filter:

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

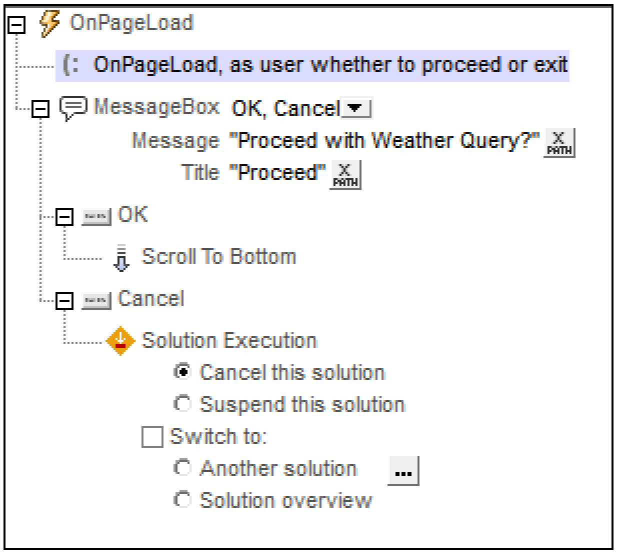
⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also* *Page Events*[389] *and* *Control Events*[665].

## 10.20.1 Comment

Comments can be added to the definition of an event's actions (*see screenshot below*). This is useful for inserting explanations of the different actions in the definition of the event's actions.

You can customize the color of comment text. To do this, click the Comment Color icon in the event pane's toolbar[669] and select a color from the color picker that appears. The selected color will be applied to all comments in all designs opened in MobileTogether Designer, including to comments that were defined prior to the setting of the new color.



OnPageLoad
(: OnPageLoad, as user whether to proceed or exit
MessageBox OK, Cancel
Message "Proceed with Weather Query?"
Title "Proceed"
OK
Scroll To Bottom
Cancel
Solution Execution
● Cancel this solution
○ Suspend this solution
□ Switch to:
○ Another solution ...
○ Solution overview

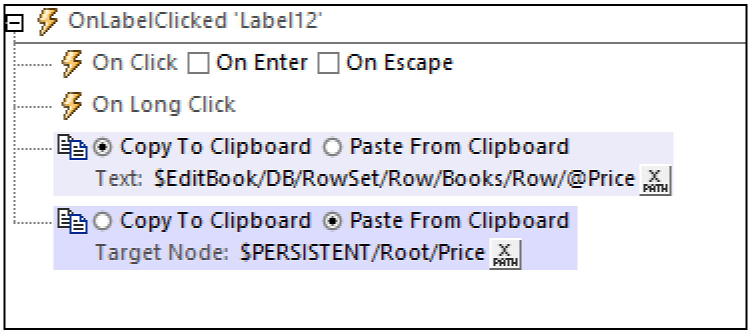
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-available-languages()`**

## 10.20.2 Copy/Paste Clipboard

You can copy text to the clipboard and subsequently paste the copied text to a target node. Each Copy/Paste action is either a copy action or a paste action: select the appropriate radio button (*see screenshot below*).



OnLabelClicked 'Label12'

On Click □ On Enter □ On Escape

On Long Click

● Copy To Clipboard ○ Paste From Clipboard
Text: $EditBook/DB/RowSet/Row/Books/Row/@Price

○ Copy To Clipboard ● Paste From Clipboard
Target Node: $PERSISTENT/Root/Price

The text that is copied to the clipboard is specified via an XPath expression. It can be static text or dynamically generated text that is based on a page source node or other contextual information.

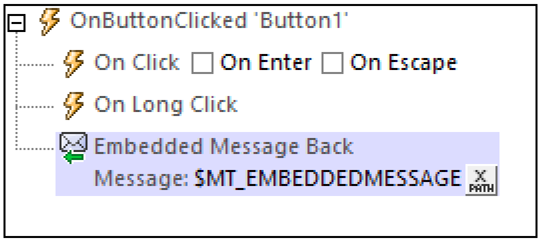
In the example of the action show in the screenshot above, the content of the **`@Price`** node is copied to the clipboard. In the subsequent Paste action, the content of the clipboard is pasted to a node of the **`$PERSISTENT`** tree. This is a simple example, where the Paste action follows immediately after the Copy action in the sequence of actions of a single event. However, the two actions can be separated in time. For example, they can each be triggered by different events. All that would required is that the Paste action follow the Copy action and that no other content be placed on the clipboard in the time that elapses between the two actions.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.20.3 Embedded Message Back

The Embedded Message Back action sends a (serialized JSON) string to the IFrame that loaded the current solution. The string is sent as a **`message`** event, and is picked up by the embedding HTML page from the IFrame by using JavaScript's `addEventListener()` method to listen for a **`message`** type event.



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

Embedded Message Back
Message: $MT_EMBEDDEDMESSAGE

The action takes as its input an XPath expression that must evaluate to a (serialized JSON) string. Any string will be accepted, but for the string to be of use in the receiving HTML page, it should be a JSON string. The XPath expression that provides the message string should therefore be one of the following:

- `$MT_EMBEDDEDMESSAGE`, which is the page source tree that contains the JSON data to be processed and transmitted. Note that the root element of this tree is always named `json`. If the entire `$MT_EMBEDDEDMESSAGE` tree is returned, as shown in the screenshot above, then the serialized JSON string will have the `json` property as its top-level property. Alternatively, the message can be defined to be a fragment of the `$MT_EMBEDDEDMESSAGE` page source, for example, `$MT_EMBEDDEDMESSAGE/json`. In this case, the serialized JSON string of the message will be the contents of the page source's `json` node.
- Any node that evaluates to a JSON data structure or a string that is a JSON data structure. The following is an example of a string that is a JSON data structure: **`'{ "books": { "author": "Mary Shelley", "title": "Frankenstein" }}'`**. In this serialized JSON string, the `books` property is the containing structure.

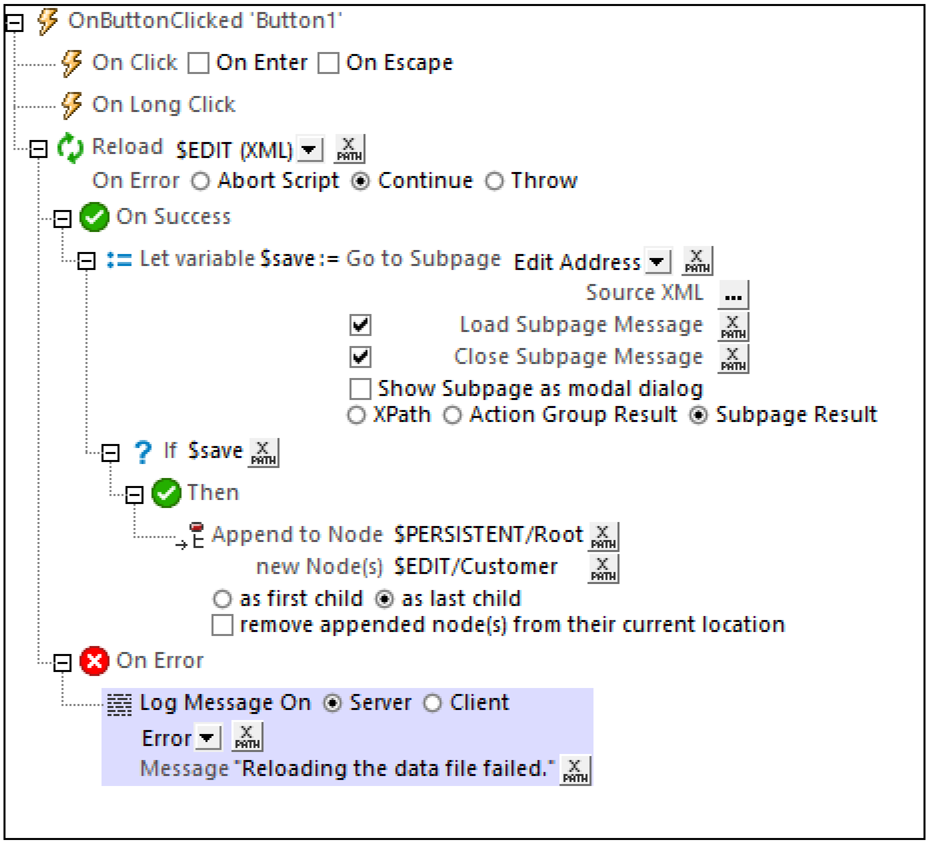
*See also*: Listening: From Server to Webpage 1436

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

## 10.20.4 Log Message

The Log Message action *(see screenshot below)* enables a customized message to be logged on the server or client during the execution of an action. This helps to analyze the app's behavior during an action. A message can be logged at any point during the execution of an action, based on the selected severity: info, warning, error.



```
OnButtonClicked 'Button1'
    On Click ☐ On Enter ☐ On Escape
    On Long Click
    Reload $EDIT (XML)
        On Error ○ Abort Script ● Continue ○ Throw
        On Success
            Let variable $save := Go to Subpage Edit Address
                                  Source XML ...
                ☑ Load Subpage Message
                ☑ Close Subpage Message
                ☐ Show Subpage as modal dialog
                ○ XPath ○ Action Group Result ● Subpage Result
            If $save
                Then
                    Append to Node $PERSISTENT/Root
                        new Node(s) $EDIT/Customer
                    ○ as first child ● as last child
                    ☐ remove appended node(s) from their current location
        On Error
            Log Message On ● Server ○ Client
                Error
                Message "Reloading the data file failed."
```

The action takes as its input:

- The location where the message is to be logged: server or client
- The severity of the error: info, warning, error
- The message to be logged, which must be entered as an XPath string

**Note:** The log messages that will be displayed also depends on the log level detail you specify in the settings of MobileTogether Server.

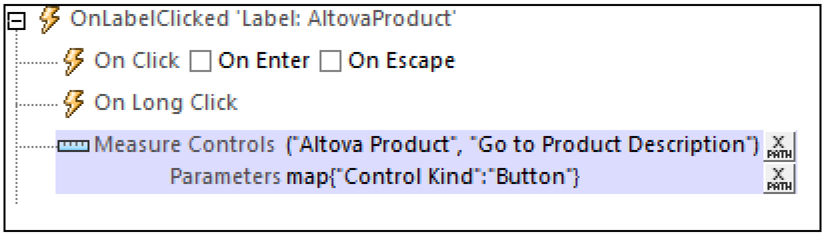
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-available-languages()`**

## 10.20.5 Measure Controls

The Measure Controls action *(screenshot below)* returns the minimum width in pixels of the specified control kind (button or label, for example) when the string specified in the first setting is the display-text of the control. The action can be used to find control widths of a set of strings and to then use this information in your design. For example, you could find the width of all buttons in a column and then set the width of this column according to the largest button-width.



⊟ OnLabelClicked 'Label: AltovaProduct'
On Click □ On Enter □ On Escape
On Long Click
Measure Controls ("Altova Product", "Go to Product Description")
Parameters map{"Control Kind":"Button"}

*Text strings of controls*
Since the control width is dependent on the size of its text, you must provide in the first setting the string that you want to place on the control. If you want to find out the widths of multiple controls, each having a different text, then supply all these texts as the string items of a sequence. The return value will be a sequence containing the widths of each string. This sequence will be stored in the **`MT_MeasureControls`** variable. For example, in the screenshot above, we have supplied a sequence of two strings: `("Altova Product", "Go to Product Description")`. These strings do not actually have to be present on controls in the design. The return value will contain two number values, which are respectively the widths of the two strings. You could add any string to the sequence to find out the control width for that string.

*Text parameters*
The second setting must be an XPath map expression that specifies the parameters for which the width calculation is made. The parameters include such information as whether the control being measured is a label or a button, or whether the text size is small or medium, or whether the text is bold or not. The keys of the map and their allowed values are listed below. The only mandatory key is `"Control Kind"`. If a key is not specified, then its default value is used. In the screenshot above, for example, the width is being measured for buttons on which the text appears with the default values of the respective parameters. The XPath expression is: `map{"Control Kind":"Button"}`.

| Key | Value | Default |
|---|---|---|
| `Control Kind` | `"Label"\|"Button"` | *Mandatory value* |
| `Text Size` | `"small"\|"medium"\|"large"` | `"medium"` |
| `Unit` | `"px"\|"dp"\|"sp"\|""`. See *Sizes: Pixels, DPI, DP, SP*[1312]. | `"px"` |
| `Bold Text` | `"true"\|"false"` | `"false"` |
| `Italic Text` | `"true"\|"false"` | `"false"` |
| `Underline Text` | `"true"\|"false"` | `"false"` |
| `Button Image` | Any of the `Button Image`[417] options (for example, + or - or > or Share | *None* |
| `Button Background` | `"transparent"\|"not_transparent"` | `"not_transparent"` |

The **`Parameters`** argument is a ***`key-value`*** map that defines the properties of the control. The available keys and their values are listed below. The integer that is returned is the minimum width, in pixels, of the control when the submitted **`Text`** string (first setting) is displayed with the properties specified in the **`Parameters`** argument. This value can then be used to calculate and specify other properties related to the control, such as the widths of table columns in which the control appears.

*The return value and the "MT_MeasureControls" variable*
The action returns a sequence of numbers, which is automatically stored in the **`MT_MeasureControls`** variable. You can use **`MT_MeasureControls`** in the XPath expressions of actions. Since the items in the returned sequence are numbers, you can use the **`max()`** function to find out the largest control width (for example: `max(MT_MeasureControls)`). Another useful function is the **`serialize()`** function, which can be used to display the complete sequence.

*Points to note*
Note the following points:

- For each action, you can find out the widths of one control kind only. To find out the widths of another control kind, call the action a second time with a new `Control Kind` parameter value.
- This action provides the same functionality as the **`mt-control-width()`** function. A key difference is that the function cannot be used for web-client rendering, whereas the action can be used for web clients.

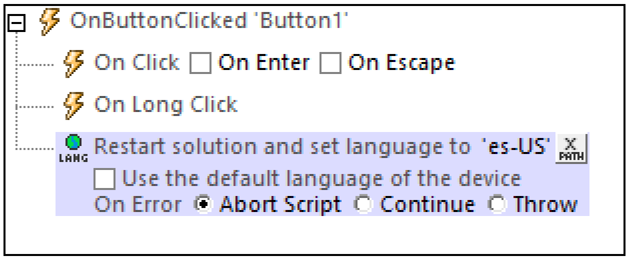
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

## 10.20.6 Set Language

If a solution is localized[308], then it can be displayed in different languages, depending on the language setting of the mobile device. For example, if a solution is designed with English text strings, then English is considered to be the solution's default language). Now, if these strings have also been localized into Spanish[1600] (that is, translated into Spanish), then the solution will automatically be displayed in English on English-language devices and in Spanish on Spanish-language devices. The key to the selection is that the language setting of the mobile device must correspond to the code name of one of the solution's localization languages.

However, you can also let the user select the language of the solution without needing to change the language of the device. The Set Language action enables the solution to be restarted with a user-selected language. For example, the user can tap a button to change to a specific language if this action is triggered by a button click (*see screenshot below, which shows an action to restart the solution in US Spanish*). Another option is to let the user select a language from a dropdown list in a combo box[459].



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

Restart solution and set language to 'es-US'

□ Use the default language of the device

On Error ● Abort Script ○ Continue ○ Throw

Enter the `language-country` code (for example, **es-US** or **fr-CH**) or just the `language` code (for example, **es** or **fr**). If the action is triggered (to select a specific language), then the solution's language will be decided according to the cascading order given below.

1. If the solution contains a matching `language-country` (**es-US** or **fr-CH**) localization, then strings of this localization are used where these exist
2. If no matching `language-country` localization (**es-US** or **fr-CH**) exists for a string, then the localized `language` (**es** or **fr**) string is used—if one exists
3. If no `language-country` localization (**es-US** or **fr-CH**) or `language` localization (**es** or **fr**) exists for a string, then the default language of the solution is used for that string

Alternatively, use the default language of the mobile device. In this case, the language setting of the device determines what language from among the available localization languages will be used. The same set of cascading rules listed above applies.

If you wish to see a simulation in any of the languages for which localized strings are defined, set the simulation language via the **Project | Simulation Language** [1606] command, and then run a simulation [1355].

*Error processing*

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable [907]. The Catch part of the Try/Catch action [907] is used to specify what action to take if an error occurs. If no error occurs, then the next action is processed. See the section Try/Catch action [907] for details.

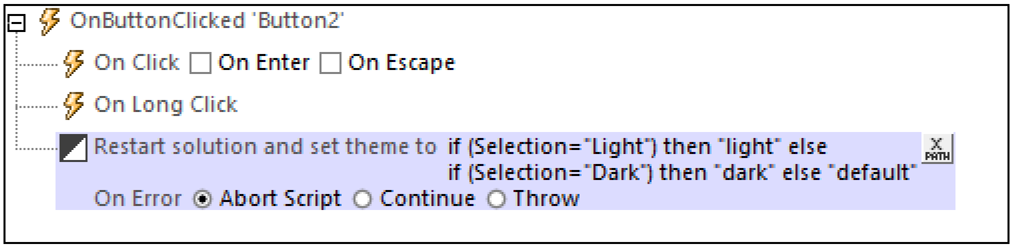
## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions [1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box [679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions [1262].

**mt-available-languages()**

## 10.20.7 Set Theme

The Set Theme action enables you to restart a solution with a new theme. This can be useful, for example, if you want to enable the end user to select the solution's theme. The theme setting that is selected by this action can be one of the following strings: **light**, **dark**, or **default**. The Set Theme action in the screenshot below shows how to use the action to select a theme if a user's choice of theme is stored in the **Selection** element.



OnButtonClicked 'Button2'
On Click □ On Enter □ On Escape
On Long Click
Restart solution and set theme to if (Selection="Light") then "light" else
if (Selection="Dark") then "dark" else "default"
On Error ● Abort Script ○ Continue ○ Throw

Note that information about the current theme is stored in the **$PERSISTENT** tree. So, if the **$PERSISTENT** tree is cleared, then the theme will have to be reset if you want to continue with the same theme.

### MobileTogether extension functions

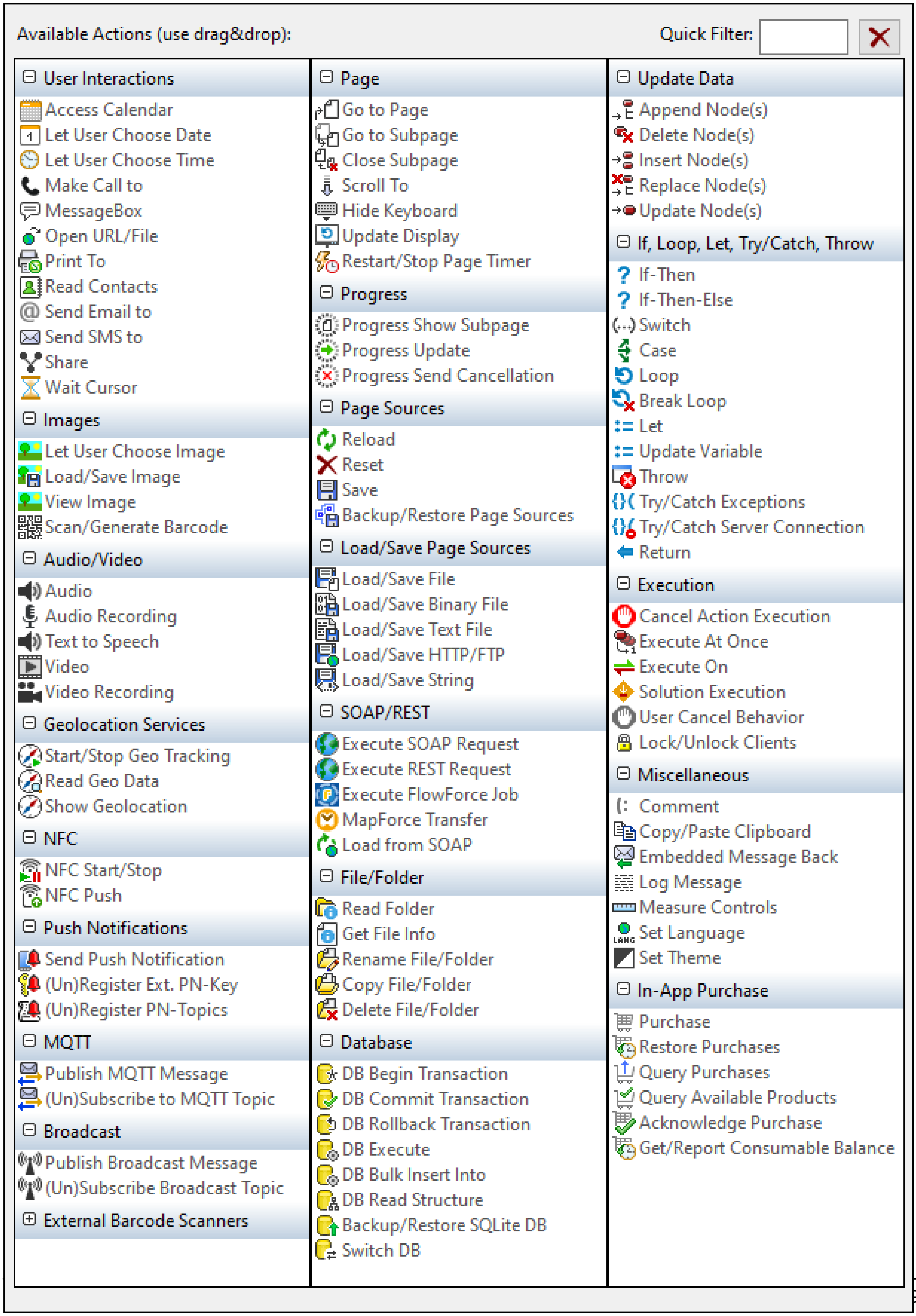
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**mt-set-theme()**

# 10.21 In-App Purchase

The following actions are available in the In-App Purchase group of the Actions dialog (*screenshot below*):

- Purchase 933
- Restore Purchases 934
- Query Purchases 934
- Query Available Products 935
- Acknowledge Purchase 936
- Get/Report Consumable Balance



Available Actions (use drag&drop): Quick Filter: ☒

⊟ User Interactions
- Access Calendar
- Let User Choose Date
- Let User Choose Time
- Make Call to
- MessageBox
- Open URL/File
- Print To
- Read Contacts
- Send Email to
- Send SMS to
- Share
- Wait Cursor

⊟ Images
- Let User Choose Image
- Load/Save Image
- View Image
- Scan/Generate Barcode

⊟ Audio/Video
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording

⊟ Geolocation Services
- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

⊟ NFC
- NFC Start/Stop
- NFC Push

⊟ Push Notifications
- Send Push Notification
- (Un)Register Ext. PN-Key
- (Un)Register PN-Topics

⊟ MQTT
- Publish MQTT Message
- (Un)Subscribe to MQTT Topic

⊟ Broadcast
- Publish Broadcast Message
- (Un)Subscribe Broadcast Topic

⊞ External Barcode Scanners

⊟ Page
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer

⊟ Progress
- Progress Show Subpage
- Progress Update
- Progress Send Cancellation

⊟ Page Sources
- Reload
- Reset
- Save
- Backup/Restore Page Sources

⊟ Load/Save Page Sources
- Load/Save File
- Load/Save Binary File
- Load/Save Text File
- Load/Save HTTP/FTP
- Load/Save String

⊟ SOAP/REST
- Execute SOAP Request
- Execute REST Request
- Execute FlowForce Job
- MapForce Transfer
- Load from SOAP

⊟ File/Folder
- Read Folder
- Get File Info
- Rename File/Folder
- Copy File/Folder
- Delete File/Folder

⊟ Database
- DB Begin Transaction
- DB Commit Transaction
- DB Rollback Transaction
- DB Execute
- DB Bulk Insert Into
- DB Read Structure
- Backup/Restore SQLite DB
- Switch DB

⊟ Update Data
- Append Node(s)
- Delete Node(s)
- Insert Node(s)
- Replace Node(s)
- Update Node(s)

⊟ If, Loop, Let, Try/Catch, Throw
- If-Then
- If-Then-Else
- Switch
- Case
- Loop
- Break Loop
- Let
- Update Variable
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection
- Return

⊟ Execution
- Cancel Action Execution
- Execute At Once
- Execute On
- Solution Execution
- User Cancel Behavior
- Lock/Unlock Clients

⊟ Miscellaneous
- Comment
- Copy/Paste Clipboard
- Embedded Message Back
- Log Message
- Measure Controls
- Set Language
- Set Theme

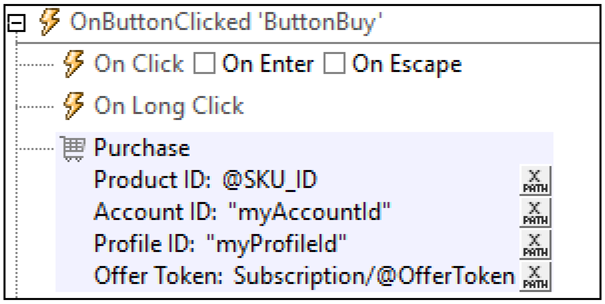
⊟ In-App Purchase
- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable Balance

The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events* 389 *and Control Events* 665.

See the section Push Notifications 1125 for an overview of how to use push notifications in MobileTogether.

## 10.21.1 Purchase

The Purchase action (*screenshot below*) sends a purchase request from the client device to the corresponding app store. The purchase request contains the ID of the product being purchased, the user's account ID and profile ID, and, for the Android store, the product's offer token if the product is a subscription. You can obtain the user IDs from, for example, a user's login credentials or some other suitable database.



OnButtonClicked 'ButtonBuy'
On Click □ On Enter □ On Escape
On Long Click
Purchase
Product ID: @SKU_ID
Account ID: "myAccountId"
Profile ID: "myProfileId"
Offer Token: Subscription/@OfferToken

When the purchase request reaches the app store, the app store will attempt to carry out the request and bill the user's account according to the user's billing information that is stored at the app store. After the app store executes the purchase request, it sends back information about the purchase. This data will be stored in the **`$MT_IN_APP_PURCHASE`** 1507 page source, with the buyer's data being stored in the **`@AccountID`** and **`@ProfileID`** attributes of the most recent **`Purchase`** element. You can authenticate the purchase by verifying that the device user's credentials match these two attributes (the buyer's credentials).

**Note:** If this action is the first In-App Purchase action 931 to be added to the design, then the `$MT_IN_APP_PURCHASE` page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source 1507.

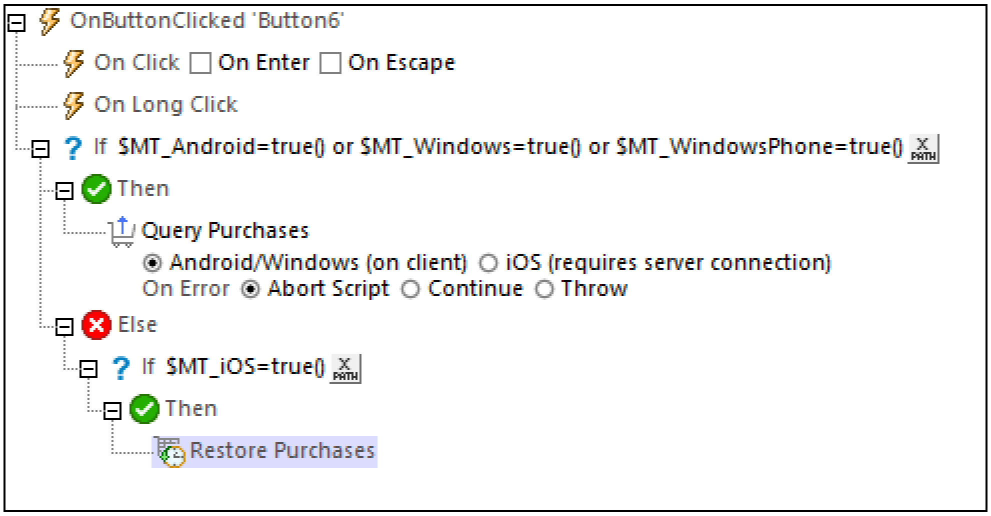
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-client-ip-address()`**
**`mt-in-app-purchase-platform-to-product()`**
**`mt-in-app-purchase-product-to-platform()`**
**`mt-in-app-purchase-service-started()`**

## 10.21.2 Restore Purchases

The Restore Purchases action (*screenshot below*) updates an iOS client with the purchases of the current user account. Purchases made by the current account, including those made from other devices, will be updated on the current device. The action updates the **`Purchases`** element of the **`$MT_IN_APP_PURCHASE`**[1507] page source. Note that this action applies only to iOS clients; or other clients, use the Query Purchases[934] action.



```
⊟ OnButtonClicked 'Button6'
    On Click □ On Enter □ On Escape
    On Long Click
  ⊟ ? If $MT_Android=true() or $MT_Windows=true() or $MT_WindowsPhone=true()
    ⊟ Then
        Query Purchases
          ● Android/Windows (on client) ○ iOS (requires server connection)
          On Error ● Abort Script ○ Continue ○ Throw
    ⊟ Else
      ⊟ ? If $MT_iOS=true()
        ⊟ Then
            Restore Purchases
```

**Note:** If this action is the first In-App Purchase action[931] to be added to the design, then the `$MT_IN_APP_PURCHASE` page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source[1507].

### MobileTogether extension functions

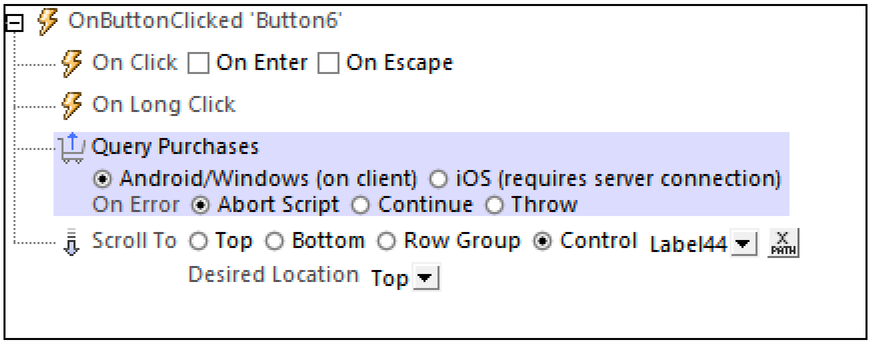
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-client-ip-address()`**
**`mt-in-app-purchase-platform-to-product()`**
**`mt-in-app-purchase-product-to-platform()`**
**`mt-in-app-purchase-service-started()`**

## 10.21.3 Query Purchases

The Query Purchases action (*screenshot below*) queries the respective app store for all purchases made with the current user account, including those made on other devices. The action updates the **`Purchases`** element of the **`$MT_IN_APP_PURCHASE`**[1507] page source.

In the case of iOS clients, a server connection is necessary because the Apple Store only provides a server-side "receipt validation". The receipt validation provides a JSON string that contains the purchase data of the current user of the device. The **`Purchases`** element of **`$MT_IN_APP_PURCHASE`**[1507] is updated with key data of the purchases returned in the JSON string. If you want to use any other data that is available in the JSON string but not added to the **`Purchases`** element (for example, the end date of a subscription), you can directly access the JSON string, which is stored in the attribute `$MT_IN_APP_PURCHASE/Root/Purchases/@OriginalJSON` (*see In-App-Purchase Page Source*[1507])[1507].



OnButtonClicked 'Button6'

On Click ☐ On Enter ☐ On Escape

On Long Click

Query Purchases

● Android/Windows (on client) ○ iOS (requires server connection)

On Error ● Abort Script ○ Continue ○ Throw

Scroll To ○ Top ○ Bottom ○ Row Group ● Control Label44

Desired Location Top

**Note:** For iOS devices that connect to the Apple Store via a server (instead of directly), additional time will be required for the additional steps and processing. So, in these cases, you should consider carefully where in the workflow you are using the Query Purchases action[934] and how this could affect performance.

**Note:** On iOS devices, you can use the Restore Purchases action[934] to obtain information about the end user's purchases..

**Note:** If this action is the first In-App Purchase action[931] to be added to the design, then the `$MT_IN_APP_PURCHASE` page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source[1507].

### MobileTogether extension functions

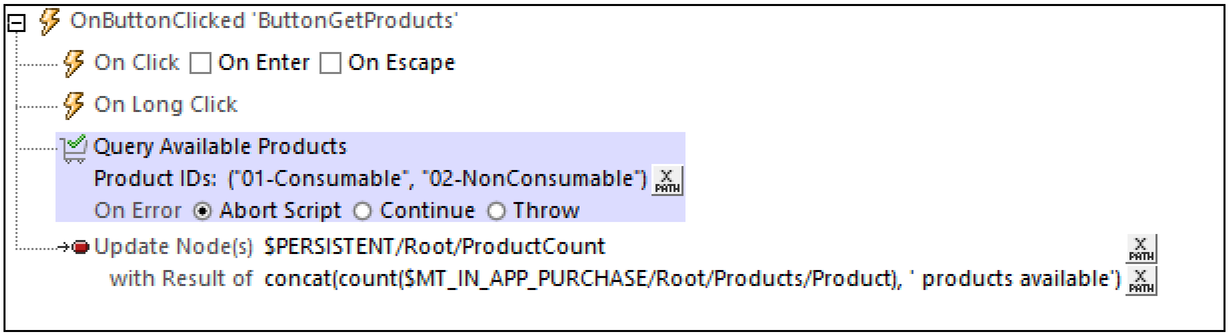
MobileTogether provides a range of XPath extension functions[1262] that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box[679] action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions[1262].

**`mt-client-ip-address()`**
**`mt-in-app-purchase-platform-to-product()`**
**`mt-in-app-purchase-product-to-platform()`**
**`mt-in-app-purchase-service-started()`**

## 10.21.4 Query Available Products

The Query Available Products action (*screenshot below*) queries the respective app store for data about the submitted product/s. The submitted products are specified in the *Product IDs* setting (*see screenshot*). The value of this setting must be a sequence of strings, each of which is a product name as defined in the In-App Purchase Products [1505]dialog. If no XPath expression is entered, then all the products defined in the In-App

Purchase Products 1505 dialog are queried. The data that is returned from the app store is stored in the **Products** element of the **$MT_IN_APP_PURCHASE** 1507 page source.



OnButtonClicked 'ButtonGetProducts'
On Click □ On Enter □ On Escape
On Long Click
Query Available Products
Product IDs: ("01-Consumable", "02-NonConsumable")
On Error ● Abort Script ○ Continue ○ Throw
Update Node(s) $PERSISTENT/Root/ProductCount
with Result of concat(count($MT_IN_APP_PURCHASE/Root/Products/Product), ' products available')

**Note:** If this action is the first In-App Purchase action 931 to be added to the design, then the `$MT_IN_APP_PURCHASE` page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source 1507.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

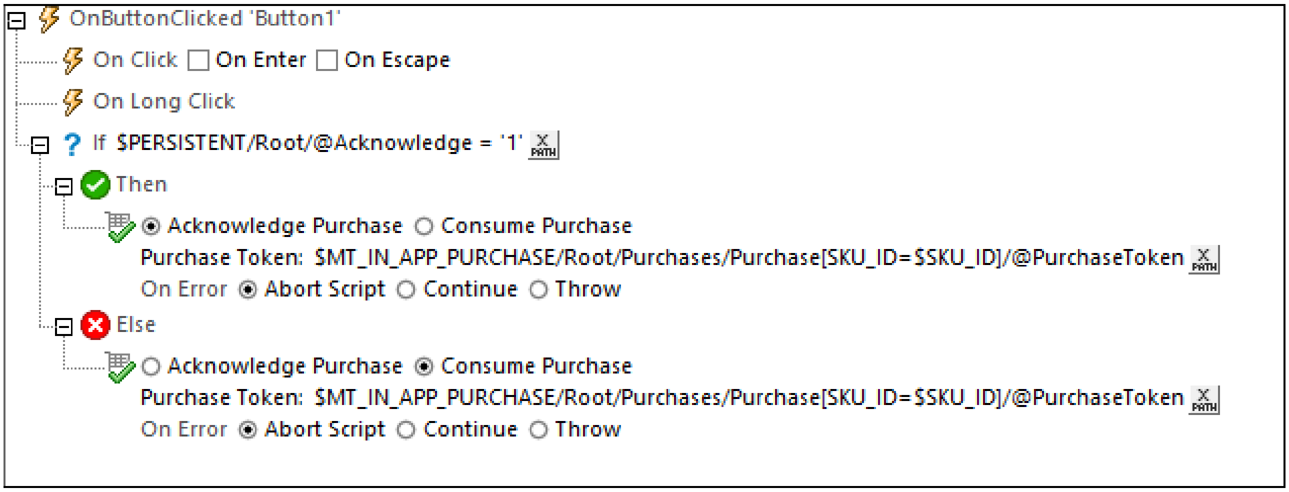
**mt-client-ip-address()**
**mt-in-app-purchase-platform-to-product()**
**mt-in-app-purchase-product-to-platform()**
**mt-in-app-purchase-service-started()**

## 10.21.5 Acknowledge Purchase

After a purchase request has been made via the Purchase 933 action, the app store attempts to process the purchase transaction. If the payment is successful, then the app store sends the purchase information to the client. It awaits an acknowledgment from the client before before finalizing the payment. If the client does not send an acknowledgment, then it could result in the purchase being canceled. Consequently, it is important to implement the Acknowledge Purchase action so that it occurs soon after the purchase information has been received from the app store. The simplest way to implement the acknowledgment action is as part of the purchasing process.

The Acknowledge Purchase action (*screenshot below*) sends this acknowledgment. The acknowledgment has to be one of two types:

- *For Non-consumables and subscriptions:* a Purchase acknowledgment.
- *For consumables:* a Consume Purchase acknowledgment.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
If $PERSISTENT/Root/@Acknowledge = '1'
Then
● Acknowledge Purchase ○ Consume Purchase
Purchase Token: $MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID=$SKU_ID]/@PurchaseToken
On Error ● Abort Script ○ Continue ○ Throw
Else
○ Acknowledge Purchase ● Consume Purchase
Purchase Token: $MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID=$SKU_ID]/@PurchaseToken
On Error ● Abort Script ○ Continue ○ Throw

The Acknowledge Purchase action has two settings:

- Select whether to: (i) Acknowledge Purchase (for non-consumables or subscriptions), or (ii) Consume Purchase (for consumables).
- The purchase token to which the action pertains. The purchase token is a unique value returned by the app store and stored in the `$MT_IN_APP_PURCHASE` page source tree, in the respective **`Purchase/@PurchaseToken`** node.

**Note:** The Acknowledge Purchase and Consume Purchase actions do not apply to Windows clients.

**Note:** If this action is the first In-App Purchase action 931 to be added to the design, then the `$MT_IN_APP_PURCHASE` page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source 1507.

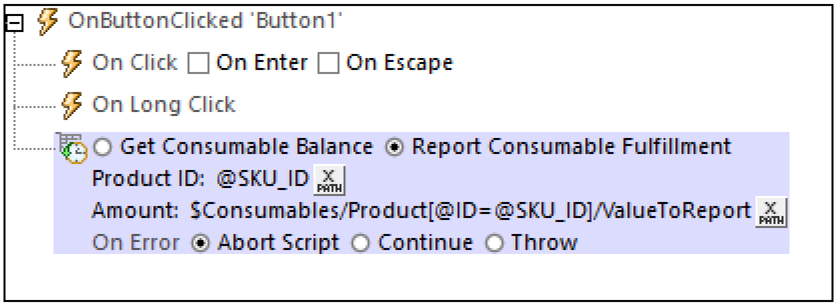
### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

**`mt-client-ip-address()`**
**`mt-in-app-purchase-platform-to-product()`**
**`mt-in-app-purchase-product-to-platform()`**
**`mt-in-app-purchase-service-started()`**

## 10.21.6 Get/Report Consumable Balance

The Get/Report Consumable action (*screenshot below*) applies to Windows devices. It enables you to either get (from the Windows app store) your current balance of consumable credit (*Get Consumable Balance*) or report your consumption of the credit to the Windows app store (*Report Consumable Fulfillment*).



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
○ Get Consumable Balance ● Report Consumable Fulfillment
Product ID: @SKU_ID
Amount: $Consumables/Product[@ID=@SKU_ID]/ValueToReport
On Error ● Abort Script ○ Continue ○ Throw

After a consumable has been purchased, it can be consumed within the app (see *Categories of InApp Purchases* 1502). In the case of Android and iOS clients, the tracking of consumable credit is handled entirely by the app on the client device. In the case of Windows clients, the Windows app store offers an option to keep a record of a user's current consumable balance. The Get/Report Consumable action addresses this Windows option. It can get the current balance from the Windows app store and report the fulfillment level to the Windows app store.

Fulfillment is different depending on how the consumable is managed:

- *Developer-managed consumable:* The consumable has been fulfilled when all of its credit units have been consumed. The app keeps track of credit consumption and reports fulfillment to the store when all credit units have been consumed. This is necessary to enable the consumable to be purchased again. If fulfillment is not reported, then a new purchase of that consumable will not be possible.
- *Store-managed consumable:* The store keeps track of credit consumption. The consumable does not need to be fully depleted before it can be purchased again. In this case, fulfillment refers to each individual incident of credit consumption. The amount consumed is reported to the store, and the store keeps track of the balance. The balance can be augmented at any time by a new purchase.

You can set up these actions at suitable points in the workflow, depending on the design of the workflow.

The following settings must be entered for the respective actions:

- *Get Consumable Balance:* the Product ID of the consumable that is being queried.
- *Report Consumable Fulfillment:* (i) the Product ID of the consumable that is being reported, (ii) either that the consumable purchase has been fulfilled (in the case of developer-managed consumables) or the amount that has been depleted (store-managed consumable).

**Note:** If this action is the first In-App Purchase action 931 to be added to the design, then the `$MT_IN_APP_PURCHASE` page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source 1507.

For more information, see Microsoft's *Overview of Consumable Add-ons*.

## MobileTogether extension functions

MobileTogether provides a range of XPath extension functions 1262 that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **`mt-available-languages()`** returns the languages in which the solution is available and could, for example, be used with the Message Box 679 action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions 1262.

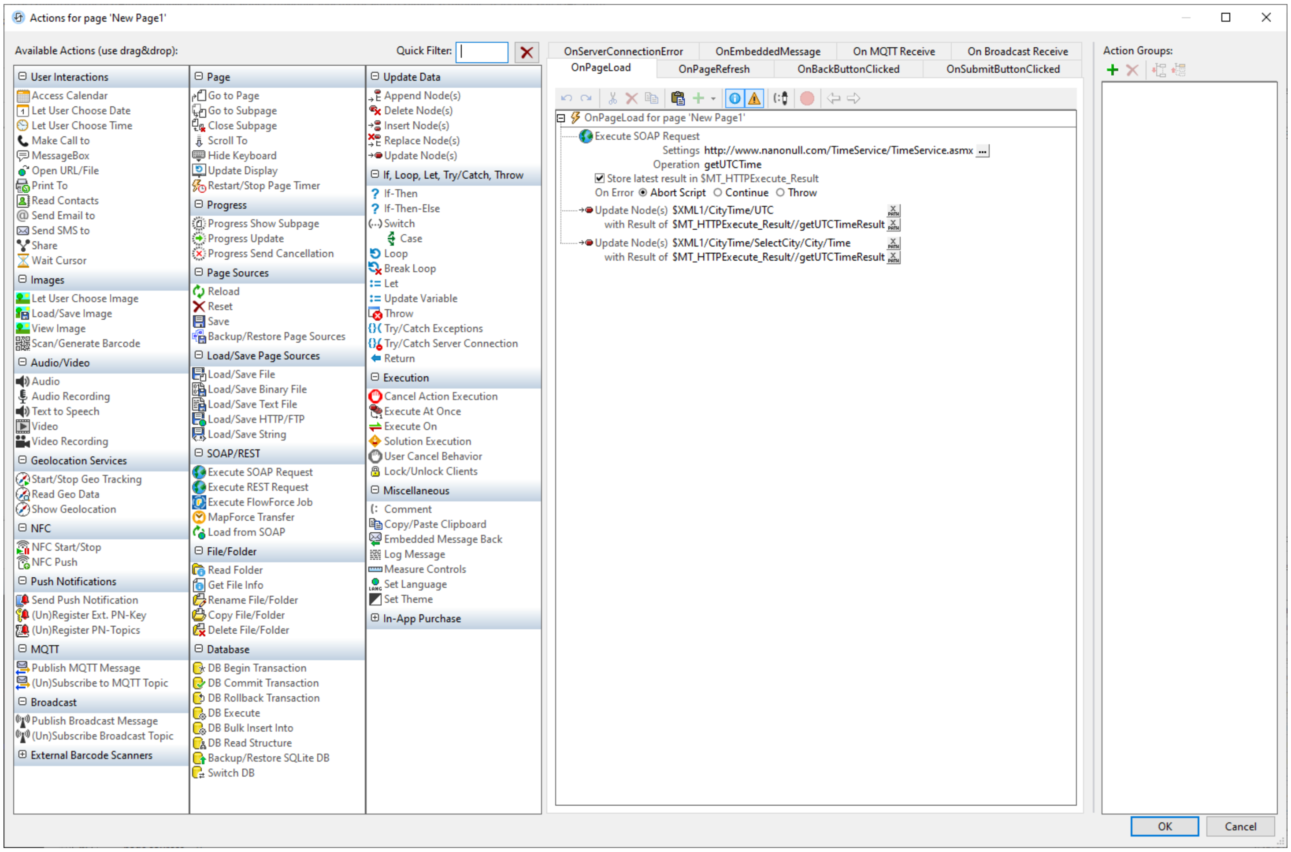
**`mt-client-ip-address()`**
**`mt-in-app-purchase-platform-to-product()`**
**`mt-in-app-purchase-product-to-platform()`**
**`mt-in-app-purchase-service-started()`**

# 10.22 Action Groups

An Action Group defines a sequence of actions to execute. If the same sequence of actions has to be executed at various points in the workflow, then it is efficient to create this sequence of actions as an Action Group and then execute this Action Group at the various workflow points. Action Groups can also have parameters, which enables you to pass dynamic values to the actions of the group.

An Action Group is created for the entire project. You can create as many Action Groups as you like. After an Action Group has been created for a project, it can be used across the pages of a project for any page event or control event of the project. You can use an Action Group as many times as you like across events.

Action Groups are managed in the right-hand side Action Groups pane of any Actions dialog (*see screenshot below*).



OnPageLoad for page 'New Page1'

Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx
Operation getUTCTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw

Update Node(s) $XML1/CityTime/UTC
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

Update Node(s) $XML1/CityTime/SelectCity/City/Time
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

This section describes:

- How to add, delete, and edit Action Groups 942
- How to use Action Groups for reusing actions 943
- How to use parameters in Action Groups 943
- How Action Groups can themselves be used as values of parameters in Action Groups 945
- How Action Groups can be used to provide the value of a variable 950 defined with the Let action 900.

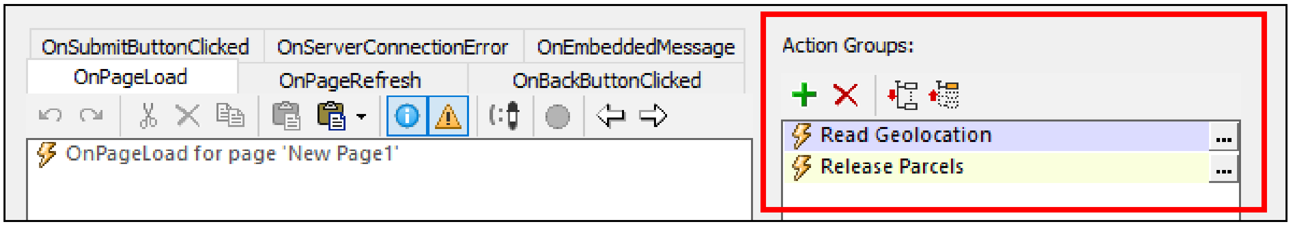
## 10.22.1 Managing Action Groups

An Action Group can be created in any Actions dialog. It is created for an entire project and is available for use by all page events and control events of the project. All the Action Groups of a project are displayed in the Action Groups pane of the Actions dialog 667, and they are managed in this pane.

### The Action Groups Pane

The Actions Group Pane (*highlighted red in the screenshot below)* enables you to do the following:

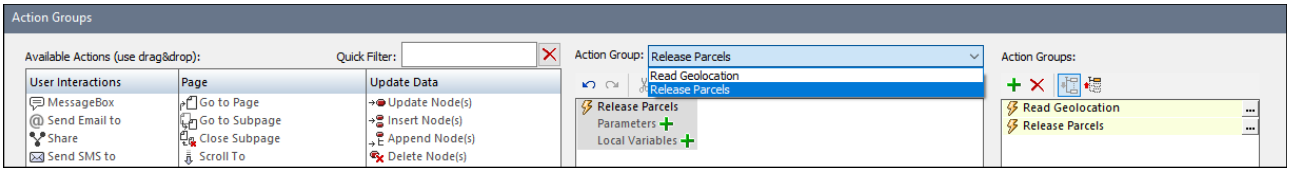
- Create an Action Group by clicking the **Add a Group** button. A new group will be added to the list of Action Groups in the pane with a generic name. You should name the group directly. Alternatively, you can edit the group's name in the Action Group's Dialog (*see next section below*) at any later time.
- Delete an Action Group by selecting it in the list and clicking the **Delete Group** button.
- Access the Action Groups Dialog (*see below*) by double-clicking an Action Group in the list of Action Groups. It will open in the main pane, where you can add actions 667 to it in the usual way.



**Note:** The *Back* and *Forward* icons in the main pane cycles through the design's Action Groups that were opened in the current session.

### The Action Groups Dialog

The Action Groups Dialog (*screenshot below*) is like any other Actions Dialog but is different in that it is used, not to define the actions of a control or a page, but those of an Action Group 940.



The Action Groups Dialog is accessed from the Action Groups Pane (*see section above*): by double-clicking an Action Group in the list of groups. The selected Action Group is opened in the Action Groups Dialog. It is the active group and can be edited in the central pane.

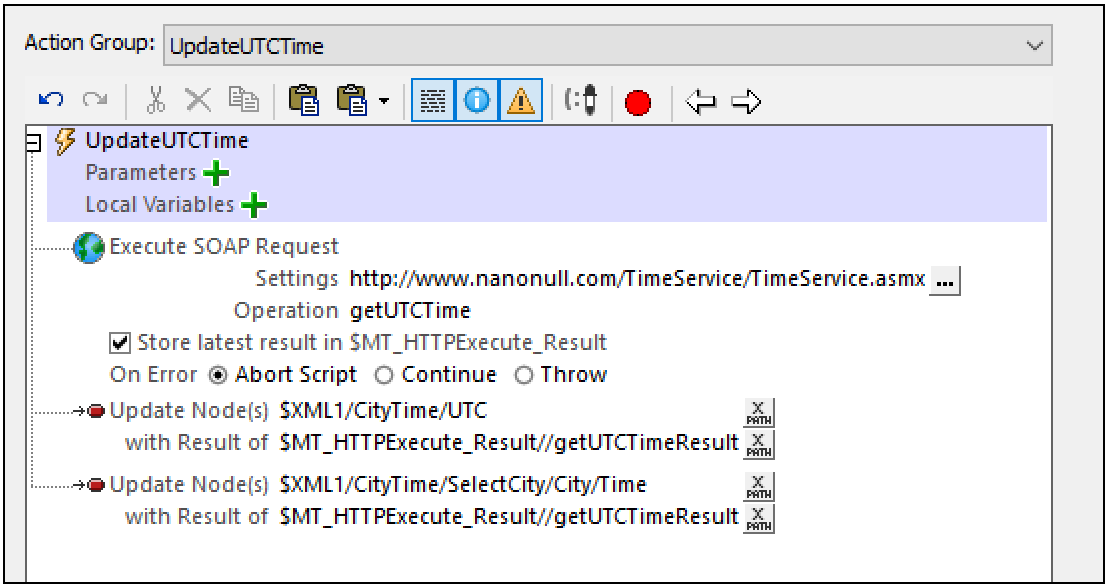
You can do the following:

- Add new actions/groups to the active group or modify existing actions/groups. Click **OK** when finished. (See the next topics of this section for details of how to work with Action Groups.)
- To edit a different Action Group, select it from the dropdown box above the central pane.
- Rename the active Action Group by double-clicking its name in the central pane and editing it.

## 10.22.2 Action Groups for Reusing Actions

Action Groups can be used to group a set of actions for reuse. Set up an Action Group for this purpose as follows:

1. Access the Action Group in order to edit it[941].
2. Drag and drop actions from the left-hand pane and specify their settings. For example, in the screenshot below, we have defined three actions for the *UpdateUTCTime* Action Group.



Action Group: UpdateUTCTime

UpdateUTCTime
Parameters
Local Variables

Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx
Operation getUTCTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw

Update Node(s) $XML1/CityTime/UTC
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

Update Node(s) $XML1/CityTime/SelectCity/City/Time
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

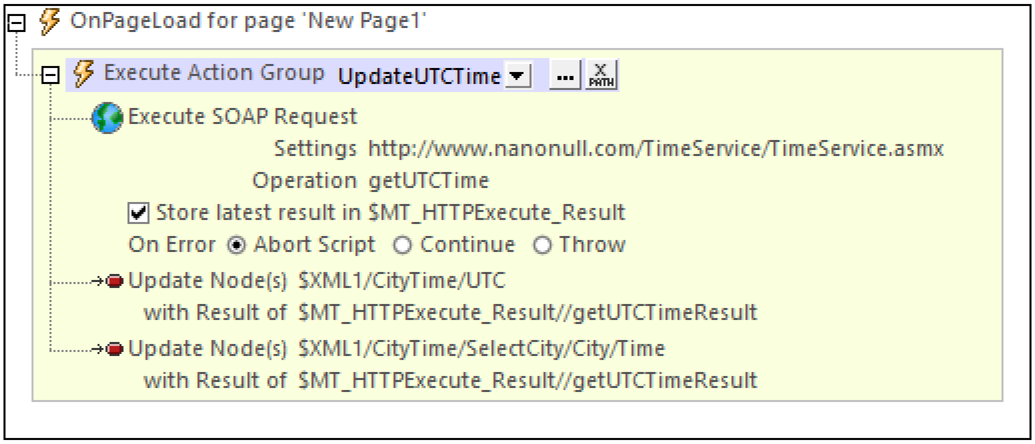
3. Click **OK** to finish.

The set of actions you have defined in the Action Group is now available for use at any point in the workflow.

**Note:** If you wish to edit the definition of another Action Group, then, in the Action Group combo box at the top of the window, select the Action Group you want to edit.

### Using the Action Group

An Action Group is used like any other action. Drag and drop it (from the Action Groups pane) into the definition of an event's actions. On being dropped, the Action Group will be displayed as an Execute Action Group action. The screenshot below shows an Execute Action Group action in its expanded form. The actions in this Action Group will be executed when the `OnPageLoad` event is triggered.



OnPageLoad for page 'New Page1'

Execute Action Group UpdateUTCTime

Execute SOAP Request

Settings http://www.nanonull.com/TimeService/TimeService.asmx

Operation getUTCTime

☑ Store latest result in $MT_HTTPExecute_Result

On Error ● Abort Script ○ Continue ○ Throw

Update Node(s) $XML1/CityTime/UTC
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

Update Node(s) $XML1/CityTime/SelectCity/City/Time
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

Note the following points:

- All the actions in the Action Group will be carried out, in the specified order, when the event is triggered.
- The Action Group can be used to execute the same set of actions at multiple points in the workflow.
- Note that the Execute Action Group action has a combo box that enables you to select any of the Action Groups defined in the project (instead of the current Action Group).
- You can also use an XPath expression to select an Action Group. The expression must evaluate to the name of the Action Group you want.
- You can click the Action Group's **Edit** button to edit the currently selected Action Group.

## 10.22.3 Action Groups with Parameters

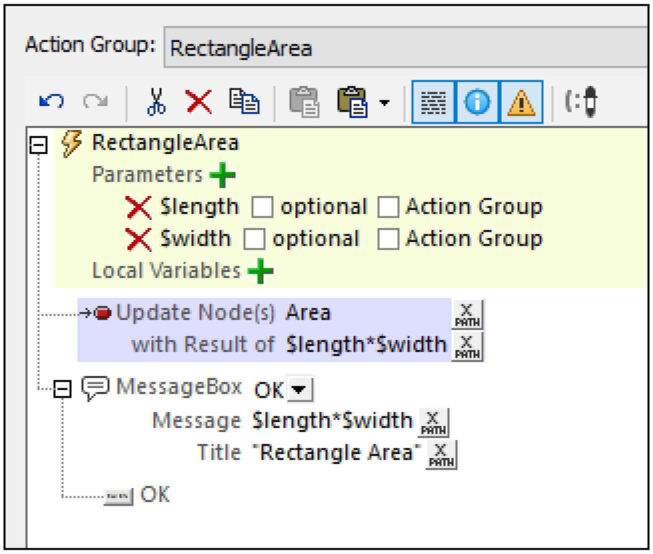
Parameters in Action Groups work as follows:

- You declare parameters in an Action Group, and then define actions (within this Action Group) that use these parameters.
- The values of parameters are passed to the parameters when the Action Group is called via the Execute Action Group action.

### Declaring parameters in Action Groups, and defining actions that use these parameters

In the Action Group, declare the parameters that are needed to generate the required result. Do this by clicking the **Add Parameter** icon (*see screenshot below*). After a new parameter has been added, double-click to the right of the parameter's **$** symbol, and enter the name of the parameter.

In the screenshot below, we have an Action Group named `RectangleArea`, in which we have declared two parameters named `$length` and `$width`. The parameters are not marked as *Optional*. This means that when the Action Group is called at runtime, the action group must receive values for both these parameters (see *Supplying parameter values with Execute Action Group* below); otherwise an error is reported. (For information about the *Action Group* check box, see the section Action Groups with Action-Group Parameters 945.)



Action Group: RectangleArea

RectangleArea
Parameters
$length □ optional □ Action Group
$width □ optional □ Action Group
Local Variables
Update Node(s) Area
with Result of $length*$width
MessageBox OK
Message $length*$width
Title "Rectangle Area"
OK

The Action Group in the screenshot above consists of two actions:

- An Update Node[886] action, which multiplies the two parameters `$length` and `$width` to generate a value that updates the `Area` node.
- A Message Box[679] action, which displays a message box containing the value obtained by multiplying the two parameters `$length` and `$width`.

Note that, in the Action Group, we declare parameters and define actions that use these parameters. The values of the parameters are supplied at runtime, via the Execute Action Group action.

## Variables in Action Groups

You can also define variables in Action Groups, that is you can declare variables and define a value for each. Note the following key points about variables:

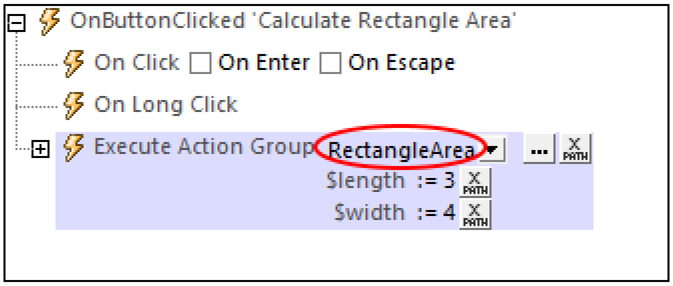
- They are in scope within the Action Group, and can only be used within the Action Group
- The definition of the value of a variable can use parameters and variables that have been declared earlier in the list of variables. For example: If an Action Group has parameters **$a**, **$b**, **$c**, and variables **$x**, **$y**, **$z** (in that order), then the variable **$y** can use the following parameters and variables to generate its value: **$a**, **$b**, **$c**, **$x**, (but not **$z**).

## Supplying parameter values with Execute Action Group

You can define an Execute Action Group action on an event, and in the action you can define the parameter values to pass to the selected Action Group. When the event is triggered, the parameter values are passed, and the Action Group uses these values to carry out the actions defined in the Action Group.

To create an Execute Action Group action, drag and drop the relevant Action Group (from the Action Groups pane[941]) into the definition of an event's actions. On being dropped, the Action Group will be displayed as an Execute Action Group action. In the screenshot below, we have defined an Execute Action Group action for an `OnButtonClicked` event. We did this by dragging the `RectangleArea` Action Group below the `On Click` event. The currently selected Action Group to execute can be changed in the Execute Action Group combo box (*circled in red in the screenshot below*). Click the **Additional Options** icon to open the selected Action Group.

Alternatively, you can use an XPath expression to select the Action Group. Using an XPath expression enables you to select the Action Group dynamically (for example, by getting the name of the Action Group from a page-source node). The XPath expression must return a string that is the name of an Action Group defined in the project.



OnButtonClicked 'Calculate Rectangle Area'
On Click □ On Enter □ On Escape
On Long Click
Execute Action Group RectangleArea
$length := 3
$width := 4

If parameters have been declared in the currently selected Action Group, then these parameters are displayed in the Execute Action Group action. This is where you supply the values to pass to parameters at runtime. In the screenshot above, it can be seen that the `RectangleArea` Action Group has two parameters: `$length` and `$width`. We have entered simple static values (`3` and `4`) in the XPath expressions that are used to generate the values of these two parameters. But you can also obtain values from XML tree sources dynamically, or specify complex XPath calculations. If, in the Action Group, the parameters were declared as mandatory (not optional), then they will be displayed here in red if no values are supplied.

At runtime, the following happens:

1. When the event is triggered, the parameter values (as specified in the Execute Action Group action) are passed to the respective parameters in the Action Group.
2. The Action Group's actions are processed. Where these use the Action Group's declared parameters, the supplied parameter values are substituted.

### Types of parameter values

In the Execute Action Group action, parameter values can be of the following types, either entered directly or generated dynamically (by using XPath expressions):

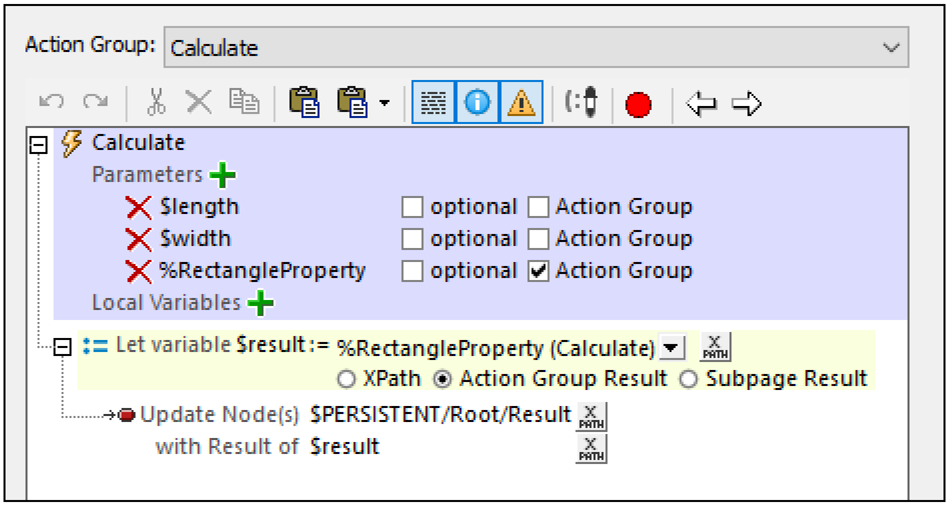
- Atomic values: for example, strings such as **`"Altova"`** or numbers such as **`1`** or **`2.56`**
- Arrays: for example, **`[(2010,2019), ("StartYear","EndYear")]`**
- Maps: for example, **`map{"StartYear":2010, "Offices":("Boston","Vienna")}`**

## 10.22.4 Action Groups with Action-Group Parameters

Action Groups can use two types of parameters:

- *Simple parameters*, which are described in the section Action Groups with Parameters 943. They are indicated by the dollar symbol **`$`** in front of their names.
- *Action-Group parameters*, which take Action Groups as their values. They are indicated by the percent symbol `%` in front of their names.

In the screenshot below, `$length` and `$width` are simple parameters, while `%RectangleProperty` is an Action-Group parameter. To create a parameter as an Action-Group parameter, create it as a simple parameter[943] and then check its *Action Group* check box (*see screenshot below*).



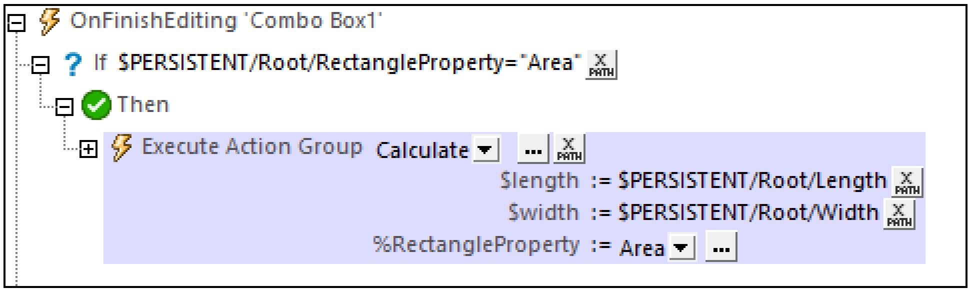
Action Group: Calculate

Calculate
Parameters
$length ☐ optional ☐ Action Group
$width ☐ optional ☐ Action Group
%RectangleProperty ☐ optional ☑ Action Group
Local Variables
Let variable $result := %RectangleProperty (Calculate)
○ XPath ● Action Group Result ○ Subpage Result
Update Node(s) $PERSISTENT/Root/Result
with Result of $result

## How Action-Group parameters work

An Action-Group parameter is a parameter whose value is an Action Group that declares no parameter. This enables the called Action Group to be treated like a function that performs MobileTogether tasks. Action-Group parameters work as follows:

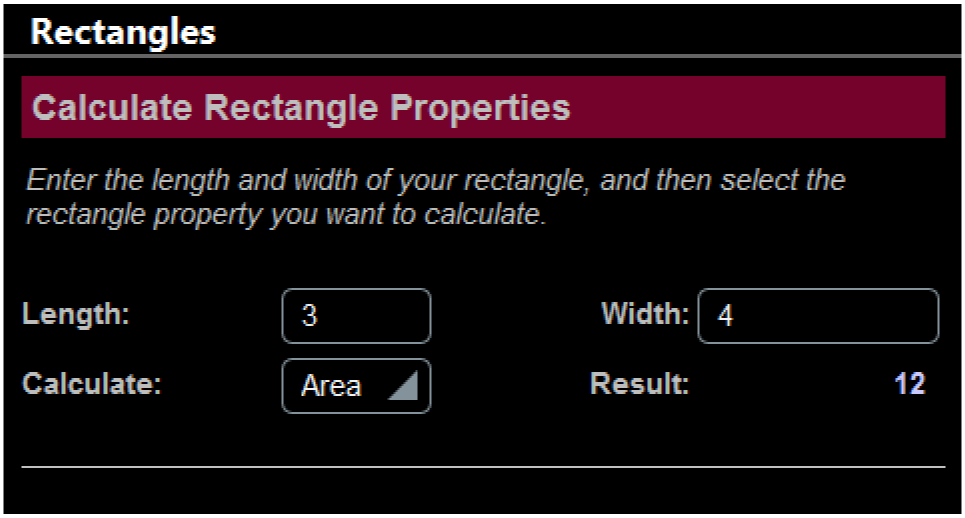
- In an Action Group, declare Action-Group parameters. These parameters can be used in actions of the Action Group. In the screenshot above, for example, the *Calculate* Action Group declares the **`RectangleProperty`** Action Group to be one of its parameters. The **`RectangleProperty`** Action Group thus becomes an Action-Group parameter. It is then used inside the *Calculate* Action Group.
- The values of all parameters of an Action Group (including those of any Action-Group parameter) are supplied when the Action Group is called. This happens when the Execute Action Group action is executed on that Action Group. For example, when we call the *Calculate* Action Group (shown in the screenshot above) via an Execute Action Group action, we pass it the values of the Action Group's three parameters *(see screenshot below)*.



OnFinishEditing 'Combo Box1'
If $PERSISTENT/Root/RectangleProperty="Area"
Then
Execute Action Group Calculate
$length := $PERSISTENT/Root/Length
$width := $PERSISTENT/Root/Width
%RectangleProperty := Area

## Usage example

Here is a simple example to show how Action-Group parameters can be used. Let us say we want to calculate the area, diagonal length, or perimeter of a rectangle from a user-given length and width *(see screenshot below)*. The user can select which rectangle property to calculate.



Rectangles

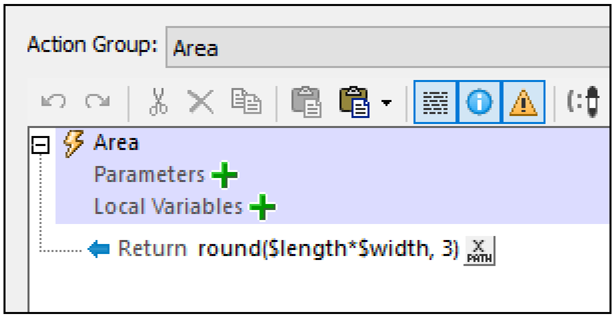
Calculate Rectangle Properties

*Enter the length and width of your rectangle, and then select the rectangle property you want to calculate.*

Length: 3 Width: 4

Calculate: Area Result: 12

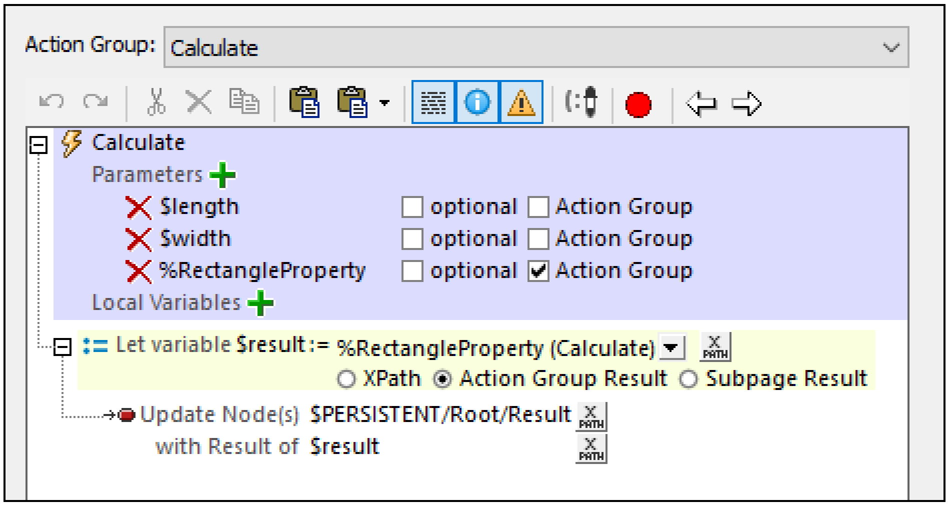
For each of the three properties, we can create an Action Group that declares no parameters. In effect, these three Action Groups are used as functions to calculate the respective rectangle property, as shown in the screenshot below.



Action Group: Area

```
Area
    Parameters
    Local Variables
  Return round($length*$width, 3)
```

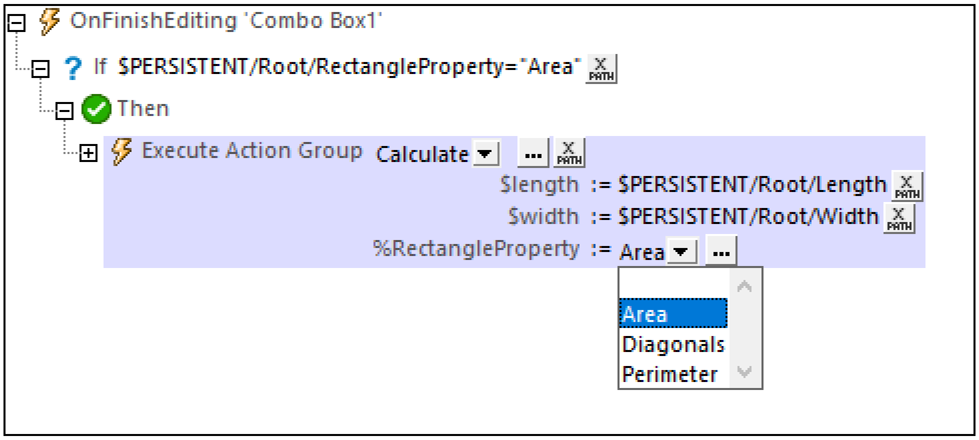
We now create a fourth Action Group (named *Calculate* in our example; *see screenshot below*) that will use one of the other three Action Groups as required. To select the Action Group e want, we create an Action-Group parameter (named **`%RectangleProperty`** in our example). Note that the *Calculate* Action Group also declares the two simple parameters (**`$length`** and **`$width`**) that are used to calculate the area, diagonals, and perimeter.



Action Group: Calculate

Calculate
Parameters
$length □ optional □ Action Group
$width □ optional □ Action Group
%RectangleProperty □ optional ☑ Action Group
Local Variables
Let variable $result := %RectangleProperty (Calculate)
○ XPath ● Action Group Result ○ Subpage Result
Update Node(s) $PERSISTENT/Root/Result
with Result of $result

All that we need to do now is to call the *Calculate* Action Group via an Execute Action Group action and, in the call, supply the values of the *Calculate* Action Group's three parameters. The Execute Action Group action is carried out ideally when some event is triggered. For example, in the screenshot below, the action is carried out when a combo box is edited.



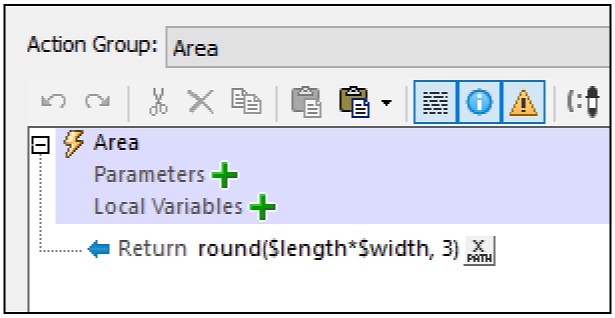
OnFinishEditing 'Combo Box1'
If $PERSISTENT/Root/RectangleProperty="Area"
Then
Execute Action Group Calculate
$length := $PERSISTENT/Root/Length
$width := $PERSISTENT/Root/Width
%RectangleProperty := Area

What happens when the Execute Action Group action is carried out is this:

1. The *Calculate* Action Group is called. Values of its **`$length`** and **`$width`** parameters are passed from nodes of the **`$PERSISTENT`** page source *(see screenshot above)*. The values of the **`RectangleProperty`** Action-Group parameter must be an Action Group in the project and can therefore be selected via the combo box; any one Action Group (corresponding to a chosen rectangle property) can be selected. In the screenshot of our example *(above)*, we have selected the *Area* Action Group (in the event that the **`$PERSISTENT/Root/RectangleProperty`** node contains the string **`"Area"`**).
2. The *Calculate* Action Group is now executed with the parameter values passed to it. If you look at its screenshot above, you will see that it defines a variable named **`$result`** and gives it a value that is the return value of the Action Group specified by the **`%RectangleProperty`** parameter. This variable will therefore contain the value of the selected rectangle property (area, diagonal length, or perimeter).
3. The *Calculate* Action Group now updates a node in the **`$PERSISTENT`** tree with this result.

*Details of the Action Group definitions*

- **`Calculate`** (*see screenshot above*): Declares two simple parameters (`$length` and `$width`) and an Action-Group parameter (`%RectangleProperty`). The Action-Group parameter `%RectangleProperty` can take as its value any of the other three Action Groups, all of which are defined without any parameter. In the *Calculate* Action Group, we define a Let action 900 that is set to *Action Group Result*. The Let action 900 defines a variable called **`$result`**, which is defined to take as its value the result of executing the selected Action Group.
- **`Area`** (*screenshot below*): This Action Group contains a single Return action 909 that contains the product of the two simple parameters `$length` and `$width`. This Action Group contains no parameter; it can therefore be the value of `%RectangleProperty`. *XPath to calculate area:* **`round(($width*$length), 3)`**.



- **`Diagonals`**: This Action Group is similar to the *Area* Action Group. It contains a single Return action 909 that returns the length of the diagonals of a rectangle (which is the square root of the sum of the squares of the two simple parameters `$length` and `$width`; *XPath:* **`round(math:sqrt($width*$width + $length*$length), 3)`**).
- **`Perimeter`**: This Action Group is similar to the *Area* and *Diagonals* Action Groups. It contains a single Return action 909 that returns the perimeter of a rectangle (*XPath:* **`round((2*$width + 2*$length), 3)`**).

In the description of the four Action Groups above, note the following:

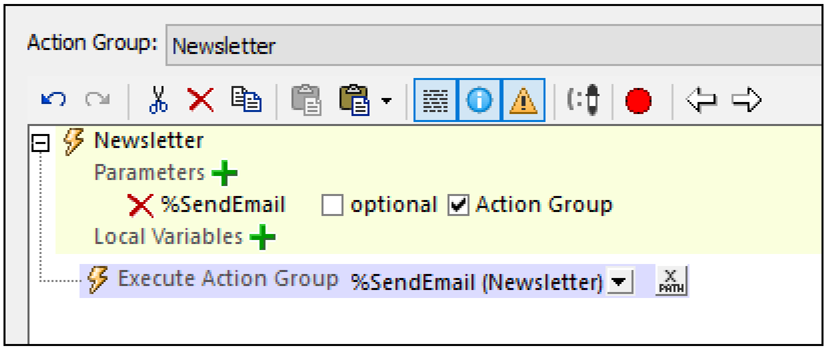
- The *Calculate* Action Group contains three parameters: two simple parameters and one Action-Group parameter.
- The other three Action Groups *(Area, Diagonals, Perimeter)* declare no parameter.
- Each of the three Action Groups *Area, Diagonals, Perimeter* has a Return action 909 that performs a calculation. The returned value is the Action Group Result.
- Each of the three Action Groups *Area, Diagonals, Perimeter* can be set as the value of the Action-Group parameter `%RectangleProperty`, and would return their respective Action Group Result when processed.

**Note:** The example above is deliberately simple; it aims to show the mechanism behind Action-Group parameters. Action-Group parameters, however, are best used with dynamic content and to execute complex actions.

## Processing of Action-Group parameters

An Action-Group parameter takes an Action Group as its value. This Action Group can be processed as a parameter value in two useful ways:

- It can provide a result—the Action Group Result[950]—which can then be used to set the value of a variable that is defined with a Let action[900]. This usage has been described in the example above.
- It can perform certain MobileTogether tasks, such as updating nodes or sending an email, and there is no Action Group Result. This is specified by defining, within the containing Action Group, an Execute Action Group action for the Action-Group parameter (*see screenshot below*). This is done by dragging the Action-Group parameter from the Action Groups pane into the event's definition. This kind of usage enables you to reuse an Action Group to perform repetitive tasks. The repetitive task can be defined in an Action Group, and the Action Group can be executed at different points during solution execution.

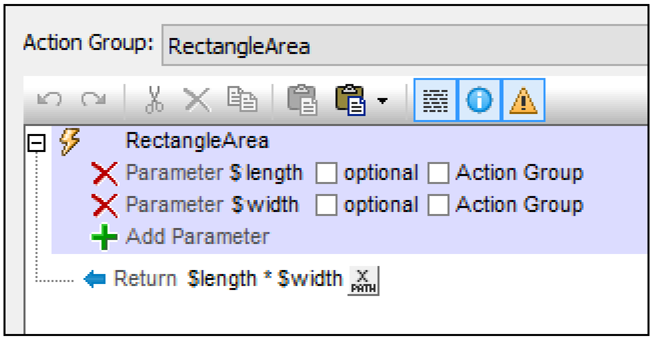

Action Group: Newsletter

Newsletter
Parameters
%SendEmail □ optional ☑ Action Group
Local Variables
Execute Action Group %SendEmail (Newsletter)

You could of course combine both sets of actions.

## 10.22.5 Variables and Action Group Results

You can set up an Action Group to return a value—the Action Group Result. (For example, in the screenshot below, we have declared two parameters, and then multiplied them in a Return action[909] to produce the Action Group Result.) When a Let action[900] is defined, we can assign the Action Group Result to a variable and use the variable in child actions of the Let action[900].


Action Group: RectangleArea

RectangleArea
Parameter $length □ optional □ Action Group
Parameter $width □ optional □ Action Group
Add Parameter
Return $length * $width

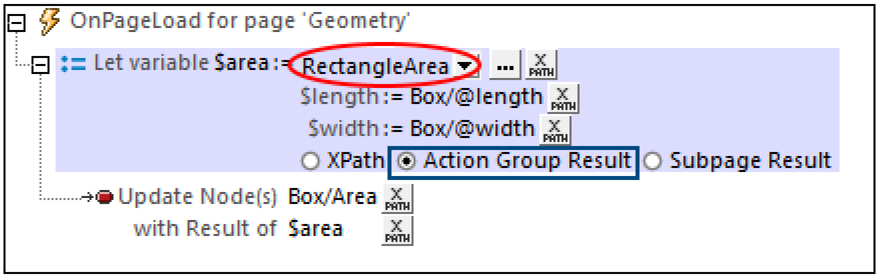
**Note:** If a variable contains a nodeset and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. However, this does not apply if only the values of variables in a nodeset have been modified.

The steps to carry out for this usage are as follows:

1. In the Action Group, declare any parameters that may be needed to generate the required result. Do this by clicking the **Add Parameter** icon (*see screenshot above*). After a new parameter has been added, double-click to the right of the parameter's **$** symbol, and enter the name of the parameter. In

the screenshot above, we have an Action Group named `RectangleArea`, in which we have declared two parameters named `$length` and `$width`. Note that: (i) the parameters have been declared but no values have been defined for them; (ii) the parameters are in scope only within the Action Group, and cannot be used outside it. If a parameter is defined as optional, then it is not an error if it is not used in the definition of the variable (*see Step 3 below*).

2. Add a Return action [909]. Use an XPath expression to define the result to return. This result will be the Action Group Result that the Let action [900] can use. In the screenshot above, we have defined an expression that multiplies the values of the `$length` and `$width` parameters. Note that the Return action [909] is within the Action Group. Consequently, the parameters are in scope.
3. To declare that a variable defined by a Let action [900] will have the value of an Action Group Result, define the Let action [900] as follows (*see screenshot below*): (i) Double-click to the right of the variable's **$** symbol, and enter the name of the variable; (ii) Select the Action Group Result radio button (*marked in blue in the screenshot below*); (iii) In the combo box at the top (*circled in red*), select the Action Group that you want to use for the value of the variable; (iv) The parameters of the selected Action Group are listed (in red if mandatory, black if optional); enter the XPath expressions to generate their values. At runtime, these values will be passed to the Action Group's parameters and used to calculate the Action Group Result.



OnPageLoad for page 'Geometry'
Let variable $area := RectangleArea
$length := Box/@length
$width := Box/@width
○ XPath ● Action Group Result ○ Subpage Result
Update Node(s) Box/Area
with Result of $area

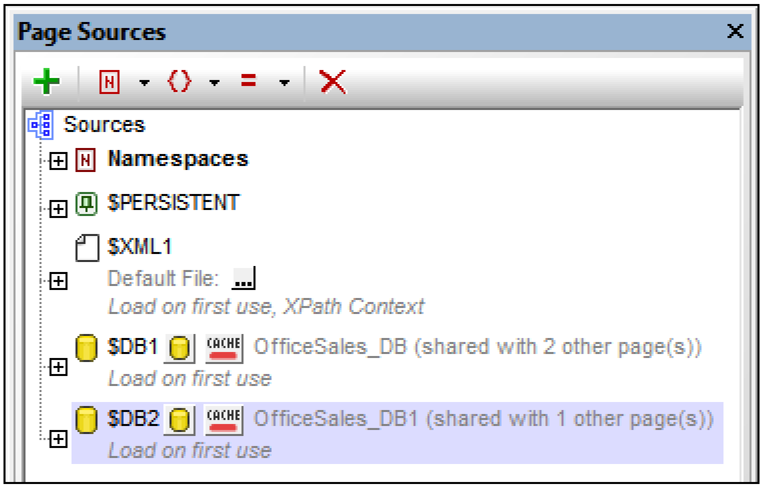
In the screenshot above, we have given, in the Let action [900], the variable a name of `$area`, and selected the `RectangleArea` Action Group to provide the variable with a value. For the parameter values, we have selected two XML tree nodes to provide the values of the `$length` and `$width` parameters (which were declared in the `RectangleArea` Action Group). When the Let action [900] is executed, the parameter values are passed to the Action Group, where the Return action [909] uses the parameter values in its calculation. The result is returned and becomes the value of the variable defined in the Let action [900]. In our example above, the values of the `$length` and `$width` parameters are passed to the `RectangleArea` Action Group, which multiplies them together and returns the result to the `$area` variable of the Let action [900].

4. The variable defined in the Let action [900] can now be used in child actions of the Let action [900]. In the screenshot above, for example, we have used the `$area` variable to update the **`Box/Area`** node.

See the description of the Let action [900] for more information.

# 11 Databases

You can use databases (DBs) as data sources of MobileTogether designs. This allows data from DBs to be displayed in MobileTogether solutions. It also allows end users to edit data in DBs from their mobile devices. You can use multiple editable DB data sources. Data in these DB sources can then be retrieved 1024, edited 1031, and saved 1035 using a variety of mechanisms, including XQuery expressions.


Page Sources
Sources
Namespaces
$PERSISTENT
$XML1
Default File:
Load on first use, XPath Context
$DB1
OfficeSales_DB (shared with 2 other page(s))
Load on first use
$DB2
OfficeSales_DB1 (shared with 1 other page(s))
Load on first use

## This section

This section is organized as follows:

- DBs as Data Sources 955
- Connecting to a DB 960
- Selecting DB Objects as Data Sources 1024
- Editing DB Data 1031
- Saving Data to the DB 1035
- The DB Execute Action 1040
- Displaying DB Data 1044
- Database Query 1046

See the database tutorial, Database-And-Charts 159, for a detailed description of a MobileTogether design which uses multiple data sources that can be edited in the MobileTogether Client solution. Also see the video demos of database use.

## Database support

The table below lists all the supported databases. If your Altova application is a 64-bit version, ensure that you have access to the 64-bit database drivers needed for the specific database you are connecting to.

| Database | Notes |
|---|---|
| Firebird 2.x, 3.x, 4.x | |

| Database | Notes |
|---|---|
| IBM DB2 8.x, 9.x, 10.x, 11.x, 12.x | |
| IBM Db2 for i 6.x, 7.4, 7.5 | Logical files are supported and shown as views. |
| IBM Informix 11.70 and later | |
| MariaDB 10 and later | MariaDB supports native connections. No separate drivers are required. |
| Microsoft Access 2003 and later | You can connect to an Access 2019 database from Altova products (i) only if the corresponding version of Microsoft Access Runtime is installed and (ii) only if the database does not use the "Large Number" data type. |
| Microsoft Azure SQL Database | SQL Server 2016 codebase |
| Microsoft SQL Server 2005 and later<br>Microsoft SQL Server on Linux | |
| MySQL 5 and later | MySQL 5.7 and later supports native connections. No separate drivers are required. |
| Oracle 9i and later | |
| PostgreSQL 8 and later | PostgreSQL connections are supported both as native connections and driver-based connections through interfaces (drivers) such as ODBC or JDBC. Native connections do not require any drivers. |
| Progress OpenEdge 11.6 | |
| SQLite 3.x | SQLite connections are supported as native, direct connections to the SQLite database file. No separate drivers are required. |
| Sybase ASE 15, 16 | |
| Teradata 16 | |

For connecting to a SQLite DB, use MobileTogether Designer's Connection Wizard 961.

Note

On Linux servers, the only database connections supported are JDBC.

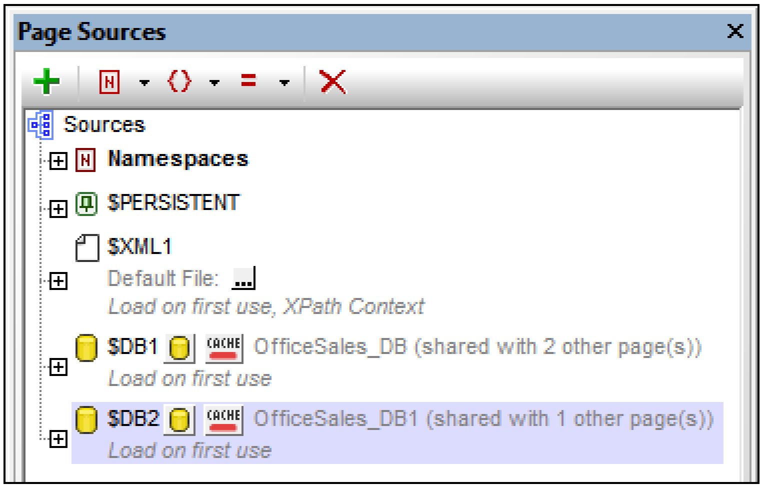
# 11.1 DBs as Data Sources

*This section:*

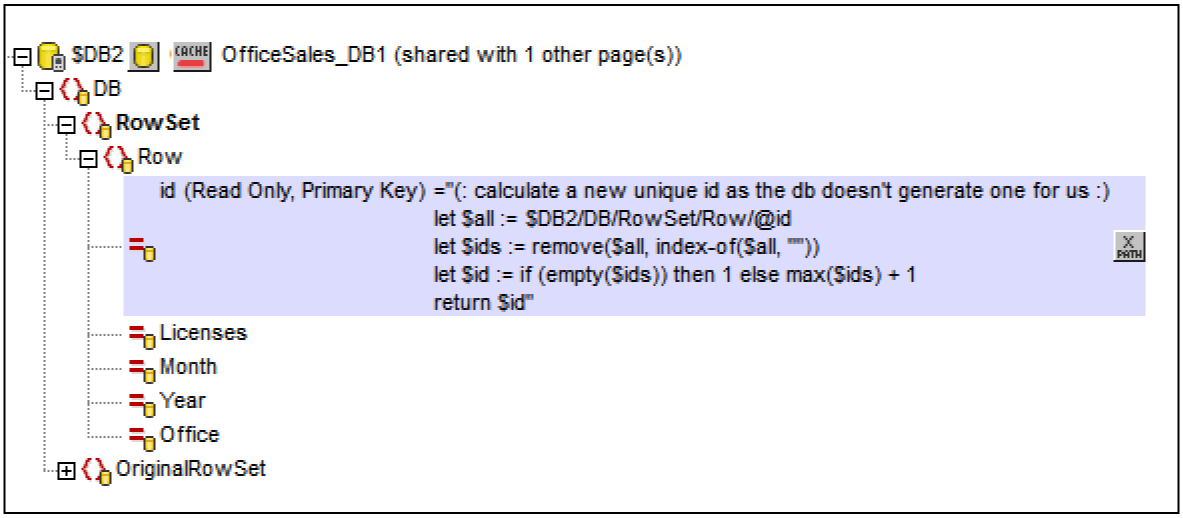
- About DB data sources 955
- Tree structure of the DB data source 956
- Switching data sources 956
- About OriginalRowSet 957
- Primary Keys in MobileTogether Designer 957

## About DB data sources

Any number of DB data sources can be added as page sources to the design of a page and then used within it. Whether a DB page source is editable or not is defined at the time the page source is added 317. Specify a DB page source to be uneditable if its data is required only for presentation. Make the page source editable if you want to allow clients to modify DB data.


Page Sources
Sources
Namespaces
$PERSISTENT
$XML1
Default File:
Load on first use, XPath Context
$DB1
CACHE
OfficeSales_DB (shared with 2 other page(s))
Load on first use
$DB2
CACHE
OfficeSales_DB1 (shared with 1 other page(s))
Load on first use

When a DB source is added, a data tree is generated (*see screenshot below and the section Tree structure below* 956). Each DB table row corresponds to a `Row` element; the table's columns are added as attributes of the `Row` element. If the page source is used on multiple pages, then a single tree structure can be shared across all instances of the page source. The option to share a tree structure is available each time a page source is added that is used on another page. When a shared structure is modified, you are offered the option of modifying the shared page source in its multiple instances across pages; alternatively, the shared page source is modified only in the instance in which it is modified..



```
$DB2  OfficeSales_DB1 (shared with 1 other page(s))
  DB
    RowSet
      Row
        id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                                      let $all := $DB2/DB/RowSet/Row/@id
                                      let $ids := remove($all, index-of($all, ""))
                                      let $id := if (empty($ids)) then 1 else max($ids) + 1
                                      return $id"
        Licenses
        Month
        Year
        Office
    OriginalRowSet
```

**Note:** If SQL statements are stored in a page source, they might trigger firewall rules while the design is executed on a client device. To prevent this, it is recommended that you do one of the following: (i) set the page source property *Keep data on* to `Server only`; (ii) use SSL for client connections; (iii) assemble the SQL statement on the server when required..

## Switching page sources

After you have created a design that uses a DB page source, you can switch the page source to another DB that has the same structure and continue to use the original design. To switch the DBs of a page source, right-click the `$DB` root node of the tree, select **Choose DB Data Source** 359, and continue with the DB connection process 960.

Two DBs are considered to have the same structure if they have the same table names, same column names, and same column definitions. If the new structure is different in any way, although the connection to the DB will be made, the page source will not be updated with data from the new DB. If the DB switch is aborted, then the page source will continue to use the original DB.

**Note:** If the DBs involved in the switch have different case-sensitivities, then you will have to modify SQL statements, XPath expressions, and any other constructs that use the non-matching names.

## Tree structure of the DB page source

Every DB page source has the following structure:

```
$DBX (the root node)
|
|--DB (the root element)
|  |
|  |--RowSet (a container element for the rows of the DB table)
|  |  |
|  |  |--Row (the rows of the DB table)
|  |  |  |
|  |  |  |--<Attributes> (the columns of the DB table)
```

When you add a DB page source[1024], you can select whether to add related tables. If a DB page source has related tables, these will be displayed as child nodes of the main table. These child nodes can be assigned to controls of the design in the usual way. When data is saved, to the DB, the child nodes are considered to belong to the hierarchy of the page source and will be saved accordingly.

The nodes in the tree can be addressed using XPath expressions. If a node is set as the *XPath context node for the page* (via the node's context menu), then XPath expressions can be built relative to the XPath context node. Otherwise nodes can be addressed using absolute paths starting at the root node: `$DBX/DB/RowSet/Row/MyAttribute`.

You can also use XQuery expressions to retrieve or manipulate data in the DB tree. See the section on primary keys below for an example.

## About **OriginalRowSet**

In order for data to be edited and saved, the tree of the page source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. The original data is saved in the `OriginalRowSet` element, while edited data is saved in the `RowSet` element. When the page source is saved, the difference between the two trees, `OriginalRowSet` and `RowSet`, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to `OriginalRowSet` so that `OriginalRowSet` contains the newly saved DB data, and the modification process can be repeated.

Note the following points:

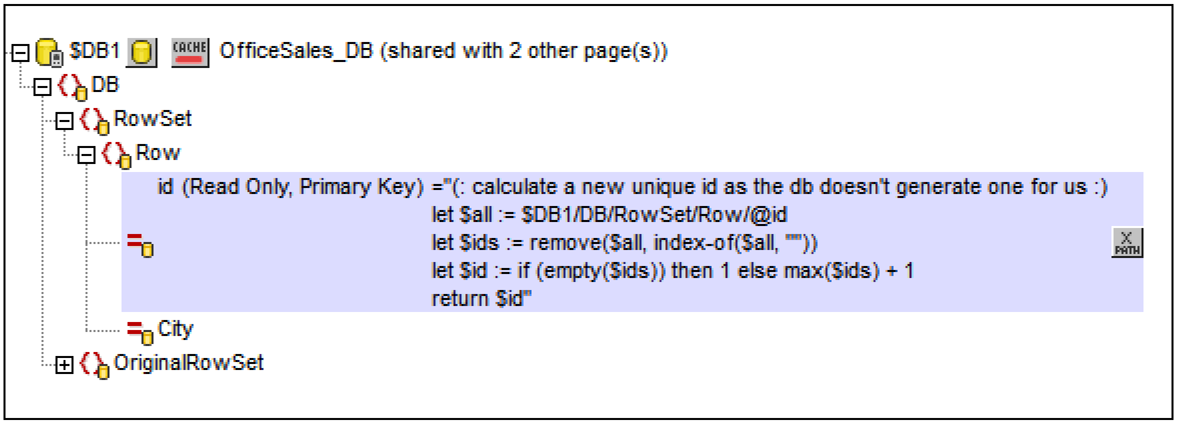
- The `OriginalRowSet` element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command **Create OriginalRowSet**.
- The **Create OriginalRowSet**.command is enabled for database type (`$DB`) root nodes. It is a toggle command that creates/removes an `OriginalRowSet` data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the `OriginalRowSet` structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function **`mt-db-original-row`**[1262].

## XPath functions for retrieving DB data or DB info

There are a number of MobileTogether XPath extension functions that can be used to retrieve DB data and information about the DB. These functions have names that begin with `mt-db` and are described in the topic MobileTogether Extension Functions[1262].

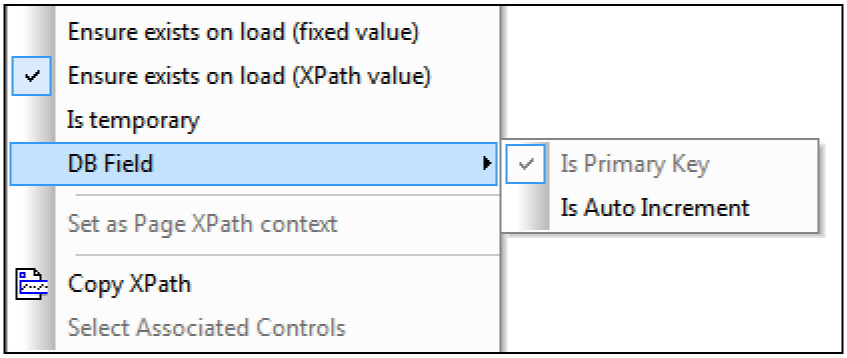
## Primary keys in MobileTogether Designer

Primary keys in DBs typically are auto-incrementing. When this is the case and a new row is added to a table, the primary key column of the added row is automatically incremented. In MobileTogether Designer, when a table is retrieved the primary key and auto-increment information is automatically retrieved and displayed in the Page Sources Pane (*see screenshot below*).



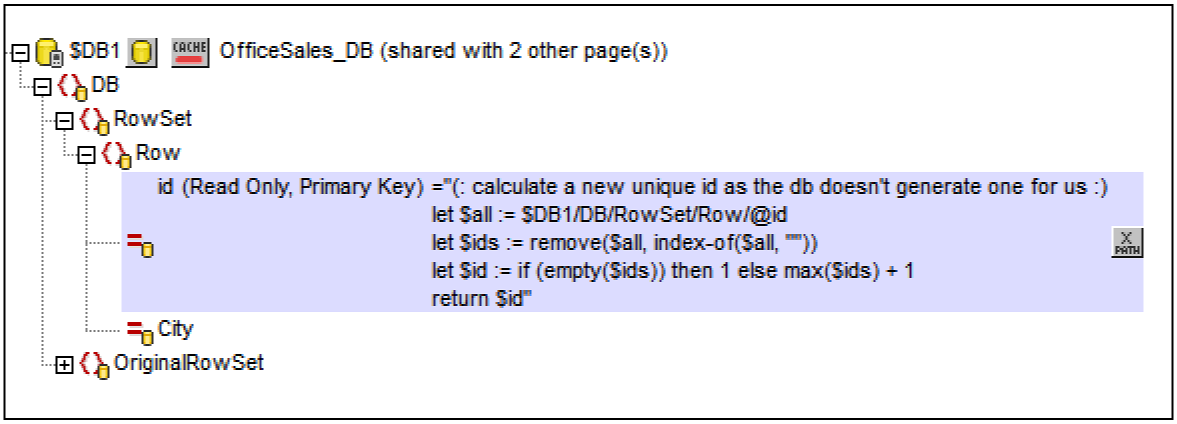
id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id"

If auto-retrieval of this information was not successful, the context menu of tree nodes contains toggle commands that enable you to correctly annotate nodes (*see screenshot below*).


Ensure exists on load (fixed value)
✓ Ensure exists on load (XPath value)
Is temporary
DB Field
✓ Is Primary Key
Is Auto Increment
Set as Page XPath context
Copy XPath
Select Associated Controls

If the primary key column is not auto-incrementing, new primary key values for appended rows must be automatically generated using an XQuery expression. This is because primary key columns cannot be edited. The XQuery expression is inserted by using the primary-key node's context menu command, **Ensure Exists before Page Load (XPath Value)**. In the example below, a new value is generated for the primary key `@id` by using the following XQuery expression:

```
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```



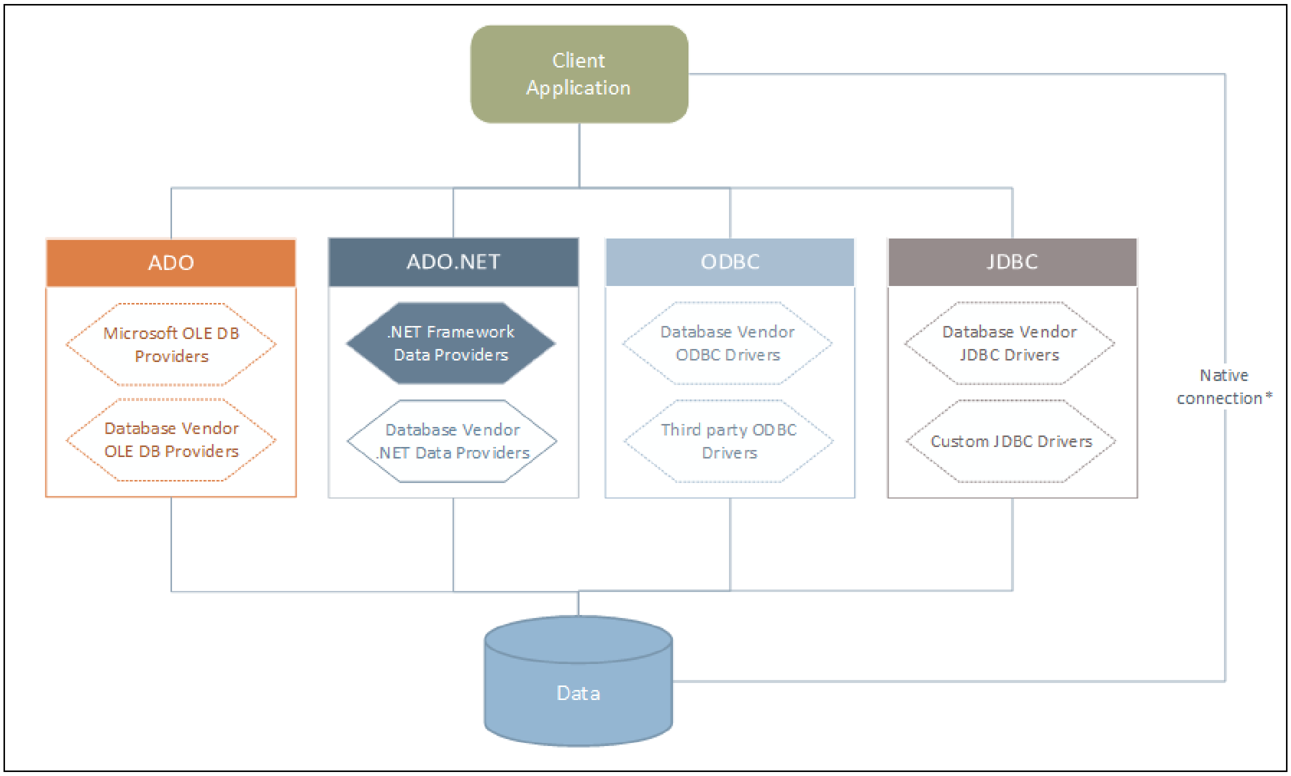
```
id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                             let $all := $DB1/DB/RowSet/Row/@id
                             let $ids := remove($all, index-of($all, ""))
                             let $id := if (empty($ids)) then 1 else max($ids) + 1
                             return $id"
```

# 11.2 Connect to a Data Source

In the most simple case, a database can be a local file such as a Microsoft Access or SQLite database file. In a more advanced scenario, a database may reside on a remote or network database server which does not necessarily use the same operating system as the application that connects to it and consumes data. For example, while MobileTogether Designer runs on a Windows operating system, the database from which you want to access data (for example, MySQL) might run on a Linux machine.

To interact with various database types, both remote and local, MobileTogether Designer relies on the data connection interfaces and database drivers that are already available on your operating system or released periodically by the major database vendors. In the constantly evolving landscape of database technologies, this approach caters for better cross-platform flexibility and interoperability.

The diagram below illustrates data connectivity options available between MobileTogether Designer (illustrated as a generic client application) and a data store (which may be a database server or database file).


Client Application
ADO
Microsoft OLE DB Providers
Database Vendor OLE DB Providers
ADO.NET
.NET Framework Data Providers
Database Vendor .NET Data Providers
ODBC
Database Vendor ODBC Drivers
Third party ODBC Drivers
JDBC
Database Vendor JDBC Drivers
Custom JDBC Drivers
Native connection*
Data

** Direct native connections 987 are supported for SQLite, MySQL, MariaDB, PostgreSQL databases. To connect to such databases, you do not need to install any additional drivers on your system.*

As shown in the diagram above, MobileTogether Designer can access any of the major database types through the following data access technologies:

- ADO (Microsoft® ActiveX® Data Objects), which, in its turn, uses an underlying OLE DB (Object Linking and Embedding, Database) provider
- ADO.NET (A set of libraries available in the Microsoft .NET Framework that enable interaction with data)

- JDBC (Java Database Connectivity)
- ODBC (Open Database Connectivity)

**Note:** Some ADO.NET providers are not supported or have limited support. See ADO.NET Support Notes 978.

### About data access technologies

The data connection interface you should choose largely depends on your existing software infrastructure. You will typically choose the data access technology and the database driver which integrates tighter with the database system to which you want to connect. For example, to connect to a Microsoft Access 2013 database, you would build an ADO connection string that uses a native provider such as the **Microsoft Office Access Database Engine OLE DB Provider**. To connect to Oracle, on the other hand, you may want to download and install the latest JDBC, ODBC, or ADO.NET interfaces from the Oracle website.

While drivers for Windows products (such as Microsoft Access or SQL Server) may already be available on your Windows operating system, they may not be available for other database types. Major database vendors routinely release publicly available database client software and drivers which provide cross-platform access to the respective database through any combination of ADO, ADO.NET, ODBC, or JDBC. In addition to this, several third party drivers may be available for any of the above technologies. In most cases, there is more than one way to connect to the required database from your operating system, and, consequently, from MobileTogether Designer. The available features, performance parameters, and the known issues will typically vary based on the data access technology or drivers used.

## 11.2.1 Start Database Connection Wizard

MobileTogether Designer provides a Database Copnnection Wizard that guides you through the steps required to set up a connection to a data source. Before you go through the wizard steps, be aware that for some database types it is necessary to install and separately configure several database prerequisites, such as a database driver or database client software. These are normally provided by the respective database vendors, and include documentation tailored to your specific Windows version. For a list of database drivers grouped by database type, see Database Drivers Overview 963.

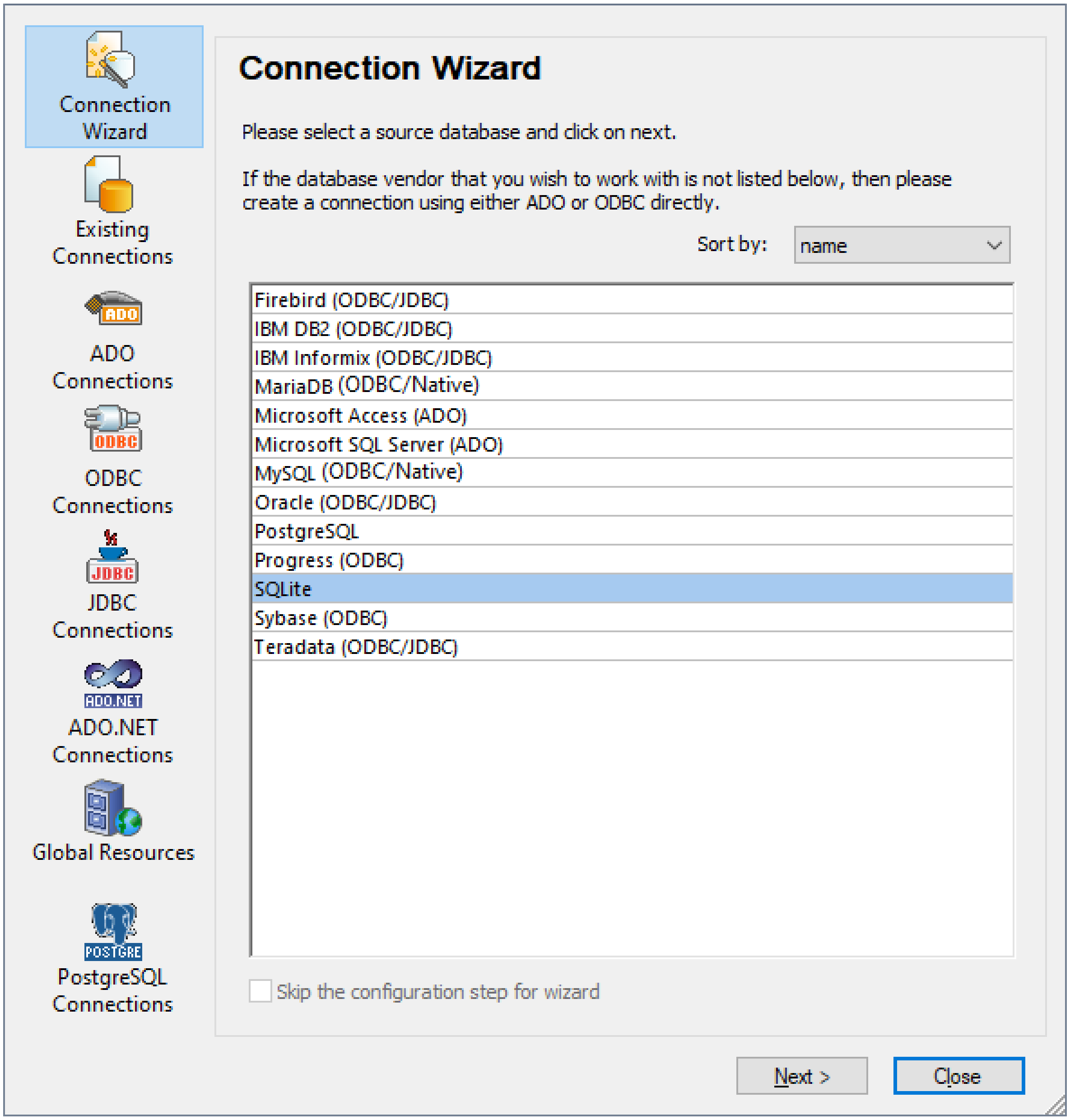
To start the Database Connection Wizard (*see screenshot below*), do the following:

- In the Page Sources Pane (of Page Design View), click the **Add Source** button and select *New DB Structure*.
- In DB Query View, click the **Quick Connect** button at the top left of the view.

The Database Connection Wizard (*screenshot below*) is started. On the left hand side of the window, you can select the most suitable from the following ways to connect to your database:

- Connection Wizard, which prompts you to choose your database type and then guides you through the steps for connecting to a database of that type
- Select an existing connection
- Select a data access technology: ADO, ADO.NET, ODBC, or JDBC
- Use an Altova global resource in which database connection is stored
- A native PostgreSQL connection

In the Connection Wizard pane (*see screenshot below*) databases can be sorted alphabetically by the name of the database type or by recent usage. Select the option you want in the *Sort By* combo box. After you have selected the database type to which you want to connect, click **Next**.



Connection Wizard

Please select a source database and click on next.

If the database vendor that you wish to work with is not listed below, then please create a connection using either ADO or ODBC directly.

Sort by: name

Firebird (ODBC/JDBC)
IBM DB2 (ODBC/JDBC)
IBM Informix (ODBC/JDBC)
MariaDB (ODBC/Native)
Microsoft Access (ADO)
Microsoft SQL Server (ADO)
MySQL (ODBC/Native)
Oracle (ODBC/JDBC)
PostgreSQL
Progress (ODBC)
SQLite
Sybase (ODBC)
Teradata (ODBC/JDBC)

☐ Skip the configuration step for wizard

The wizard will take you through the next steps according to the database type, connection technology (ADO, ADO.NET, ODBC, JDBC), and driver that will be used. For examples applicable to each database type, see Database Connection Examples 989.

Alternatively to using Connection Wizard, you can use one of the following database access technologies:

- Setting up an ADO Connection 967
- Setting up an ADO.NET Connection 973
- Setting up an ODBC Connection 982

- Setting up a JDBC Connection 979

## 11.2.2 Database Drivers Overview

This topic gives an overview of database drivers. Even though a number of database drivers might be already available on your Windows operating system, you may still need to download an alternative driver.

Database vendors may provide drivers either as separate downloadable packages, or bundled with database client software. In the latter case, the database client software normally includes any required database drivers, or provides you with an option during installation to select the drivers and components you wish to install. Database client software typically consists of administration and configuration utilities used to simplify database administration and connectivity, as well as documentation on how to install and configure the database client and any of its components.

Configuring the database client correctly is crucial for establishing a successful connection to the database. Before installing and using the database client software, we recommend that you carefully read the installation and configuration instructions of the database client; these may vary for each database version and for each Windows version.

To understand the capabilities and limitations of each data access technology with respect to each database type, refer to the documentation of that particular database product and also test the connection against your specific environment. To avoid common connectivity issues, note the following:

- Some ADO.NET providers are not supported or have limited support. See ADO.NET Support Notes 978.
- When installing a database driver, it is recommended that it has the same platform as the Altova application (32-bit or 64-bit). For example, if you are using a 32-bit Altova application on a 64-bit operating system, install the 32-bit driver, and set up your database connection using the 32-bit driver, see also Viewing the Available ODBC Drivers 984.
- When setting up an ODBC data source, it is recommended that you create the data source name (DSN) as a System DSN instead of as a User DSN. For more information, see Setting up an ODBC Connection 982.
- When setting up a JDBC data source, ensure that JRE (Java Runtime Environment) or Java Development Kit (JDK) is installed and that the **`CLASSPATH`** environment variable of the operating system is configured. For more information, see Setting up a JDBC Connection 979.
- For the installation instructions and support details of any drivers or database client software that you install from a database vendor, check the documentation provided with the installation package.

### Available data-access technologies

The table below lists common database drivers you can use to connect to a particular database through a particular data access technology. Note that this list is neither exhaustive nor prescriptive; you can use other native or third party alternatives in addition to the drivers shown below.

| Database | Interface | Drivers |
|---|---|---|
| Firebird | ADO.NET | Firebird ADO.NET Data Provider (https://www.firebirdsql.org/en/additional-downloads/) |
| | JDBC | Firebird JDBC driver ( https://www.firebirdsql.org/en/jdbc-driver/ ) |

| Database | Interf ace | Drivers |
|---|---|---|
| | | |
| | ODBC | Firebird ODBC driver ( https://www.firebirdsql.org/en/odbc-driver/ ) |
| IBM DB2 | ADO | IBM OLE DB Provider for DB2 |
| | ADO. NET | IBM Data Server Provider for .NET |
| | JDBC | IBM Data Server Driver for JDBC and SQLJ |
| | ODBC | IBM DB2 ODBC Driver |
| IBM DB2 for i | ADO | • IBM DB2 for i5/OS IBMDA400 OLE DB Provider<br>• IBM DB2 for i5/OS IBMDARLA OLE DB Provider<br>• IBM DB2 for i5/OS IBMDASQL OLE DB Provider |
| | ADO. NET | .NET Framework Data Provider for IBM i |
| | JDBC | IBM Toolbox for Java JDBC Driver |
| | ODBC | iSeries Access ODBC Driver |
| IBM Informix | ADO | IBM Informix OLE DB Provider |
| | JDBC | IBM Informix JDBC Driver |
| | ODBC | IBM Informix ODBC Driver |
| Microsoft Access | ADO | • Microsoft Jet OLE DB Provider<br>• Microsoft Access Database Engine OLE DB Provider |
| | ADO. NET | .NET Framework Data Provider for OLE DB |
| | ODBC | • Microsoft Access Driver |
| MariaDB | ADO. NET | In the absence of a dedicated .NET connector for MariaDB, use **Connector/NET** for MySQL (https://dev.mysql.com/downloads/connector/net/). |

| Database | Interface | Drivers |
|---|---|---|
| | | |
| | JDBC | MariaDB Connector/J (https://downloads.mariadb.org/) |
| | ODBC | MariaDB Connector/ODBC (https://downloads.mariadb.org/) |
| | Native connection | Available. No drivers are required. |
| Microsoft SQL Server | ADO | • Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)<br>• Microsoft OLE DB Provider for SQL Server (SQLOLEDB)<br>• SQL Server Native Client (SQLNCLI) |
| | ADO.NET | • .NET Framework Data Provider for SQL Server<br>• .NET Framework Data Provider for OLE DB |
| | JDBC | • Microsoft JDBC Driver for SQL Server ( https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server ) |
| | ODBC | • ODBC Driver for Microsoft SQL Server ( https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server ) |
| MySQL | ADO.NET | • Connector/NET (https://dev.mysql.com/downloads/connector/net/) |
| | JDBC | Connector/J ( https://dev.mysql.com/downloads/connector/j/ ) |
| | ODBC | Connector/ODBC ( https://dev.mysql.com/downloads/connector/odbc/ ) |
| | Native connection | Available for MySQL 5.7 and later. No drivers are required. |
| Oracle | ADO | • Oracle Provider for OLE DB<br>• Microsoft OLE DB Provider for Oracle |
| | ADO.NET | Oracle Data Provider for .NET (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html) |
| | JDBC | • JDBC Thin Driver<br>• JDBC Oracle Call Interface (OCI) Driver |

| Database | Interface | Drivers |
|---|---|---|
| | | These drivers are typically installed during the installation of your Oracle database client. Connect through the OCI Driver (not the Thin Driver) if you are using the Oracle XML DB component. |
| | ODBC | • Microsoft ODBC for Oracle<br>• Oracle ODBC Driver (typically installed during the installation of your Oracle database client) |
| PostgreSQL | JDBC | PostgreSQL JDBC Driver ( https://jdbc.postgresql.org/download.html ) |
| | ODBC | psqlODBC ( https://odbc.postgresql.org/ ) |
| | Native connection | Available. No drivers are required. |
| Progress OpenEdge | JDBC | JDBC Connector ( https://www.progress.com/jdbc/openedge ) |
| | ODBC | ODBC Connector ( https://www.progress.com/odbc/openedge ) |
| SQLite | Native connection | Available. No drivers are required. |
| Sybase | ADO | Sybase ASE OLE DB Provider |
| | JDBC | jConnect™ for JDBC |
| | ODBC | Sybase ASE ODBC Driver |
| Teradata | ADO.NET | .NET Data Provider for Teradata (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata) |
| | JDBC | Teradata JDBC Driver (https://downloads.teradata.com/download/connectivity/jdbc-driver) |
| | ODBC | Teradata ODBC Driver for Windows (https://downloads.teradata.com/download/connectivity/odbc-driver/windows) |

## 11.2.3 ADO Connection

Microsoft ActiveX Data Objects (ADO) is a data access technology that enables you to connect to a variety of data sources through OLE DB. OLE DB is an alternative interface to ODBC or JDBC; it provides uniform access to data in a COM (Component Object Model) environment. ADO is a precursor of the newer ADO.NET 973 and is still one of the possible ways to connect to Microsoft native databases such as Microsoft Access or SQL Server, although you can also use it for other data sources.

Importantly, you can choose between multiple ADO providers, and some of them must be downloaded and installed on your workstation before you can use them. For example, for connecting to SQL Server, the following ADO providers are available:

- Microsoft OLE DB *Driver* for SQL Server (MSOLEDBSQL)
- Microsoft OLE DB *Provider* for SQL Server (SQLOLEDB)
- SQL Server Native Client (SQLNCLI)

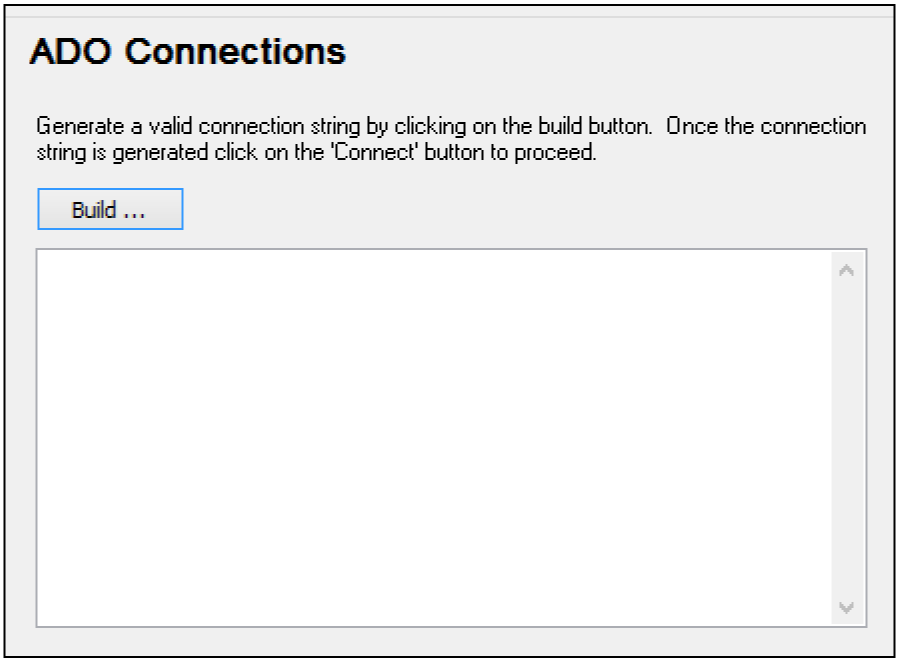
From the providers listed above, the recommended one is MSOLEDBSQL; you can download it from https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15. Note that it must match the platform of MobileTogether Designer (32-bit or 64-bit). The SQLOLEDB and SQLNCLI providers are considered deprecated and thus are not recommended.

**Note:** The *Microsoft OLE DB Provider for SQL Server (SQLOLEDB)* is known to have issues with parameter binding of complex queries like Common Table Expressions (CTE) and nested SELECT statements.

### Set up an ADO connection

To set up an ADO connection, do the following:

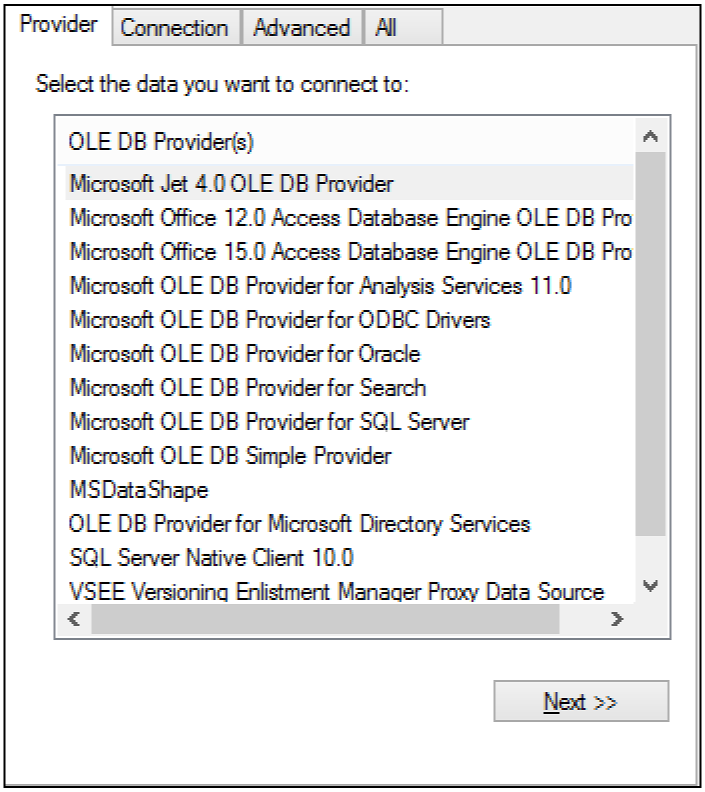
1. Start the database connection wizard 961.
2. Click **ADO Connections**.



ADO Connections

Generate a valid connection string by clicking on the build button. Once the connection string is generated click on the 'Connect' button to proceed.

3. Click **Build**.



Select the data you want to connect to:

OLE DB Provider(s)
Microsoft Jet 4.0 OLE DB Provider
Microsoft Office 12.0 Access Database Engine OLE DB Pro
Microsoft Office 15.0 Access Database Engine OLE DB Pro
Microsoft OLE DB Provider for Analysis Services 11.0
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Services
SQL Server Native Client 10.0
VSEE Versioning Enlistment Manager Proxy Data Source

4. Select the data provider through which you want to connect. The table below lists a few common scenarios.

| To connect to this database... | Use this provider... |
|---|---|
| Microsoft Access | - **Microsoft Office Access Database Engine OLE DB Provider** (recommended)<br>- **Microsoft Jet OLE DB Provider**<br><br>If the **Microsoft Office Access Database Engine OLE DB Provider** is not available in the list, make sure that you have installed either Microsoft Access or the Microsoft Access Database Engine Redistributable (https://www.microsoft.com/en-us/download/details.aspx?id=54920) on your computer. |
| SQL Server | - **Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)** - this is the recommended OLE DB provider. In order for this provider to appear in the list, it must be downloaded from https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 and installed.<br>- **Microsoft OLE DB Provider for SQL Server (OLEDBSQL)**<br>- **SQL Server Native Client (SQLNCLI)** |
| Other database | Select the provider applicable to your database.<br><br>If an OLE DB provider to your database is not available, install the required driver from the database vendor (see Database Drivers Overview 963). Alternatively, set up an ADO.NET, ODBC, or JDBC connection.<br><br>If the operating system has an ODBC driver to the required database, you could also use the **Microsoft OLE DB Provider for ODBC Drivers**, or preferably opt for an ODBC connection 982. |

5. Having selected the provider of choice, click **Next** and complete the wizard.

The subsequent wizard steps are specific to the provider you chose. For SQL Server, you will need to provide or select the host name of the database server, the authentication method, the database name, as well as the database username and password. For an example, see Connecting to Microsoft SQL Server (ADO) 1004. For Microsoft Access, you will be asked to browse for or provide the path to the database file. For an example, see Connecting to Microsoft Access (ADO) 1002.

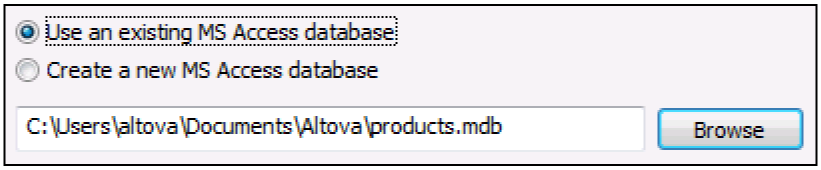
The complete list of initialization properties (connection parameters) is available in the **All** tab of the connection dialog box—these properties vary depending on the chosen provider and may need to be set explicitly in order for the connection to be possible. The following sections provide guidance on configuring the basic initialization properties for Microsoft Access and SQL Server databases:

- Setting up the SQL Server Data Link Properties 971
- Setting up the Microsoft Access Data Link Properties 971

### 11.2.3.1 Connect to an Existing MS Access Database

The procedure given below is suitable if you want to connect to a Microsoft Access database that is not password-protected. If the database is password-protected, use the method given in the Connection Example for Microsoft Access (ADO)[1002].

1. Run the database connection wizard (see Starting the Database Connection Wizard[961]).
2. Select **Microsoft Access (ADO)**, and then click **Next**.



● Use an existing MS Access database
○ Create a new MS Access database
C:\Users\altova\Documents\Altova\products.mdb

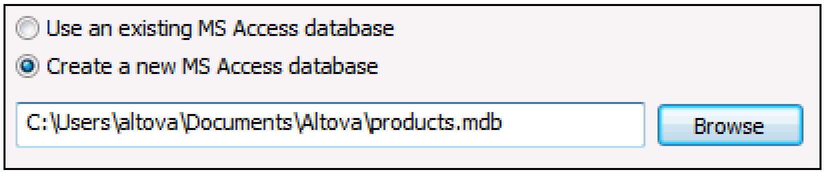
3. Select *Use an existing MS Access database*.
4. Browse for the database file, or enter the path to it (either relative or absolute).
5. Click **Connect**.

### 11.2.3.2 Create a New MS Access Database

As an alternative to connecting to an existing database file, you can create a new Microsoft Access database file (.accdb, .mdb) and connect to it. You can do this even if Microsoft Access is not installed on the computer. The database file that is created by MobileTogether Designer will be empty. To create the required database structure, use Microsoft Access or a tool such as DatabaseSpy ( https://www.altova.com/databasespy ).

Create a new MS Access database as follows:

1. Run the database connection wizard (see Starting the Database Connection Wizard[961]).
2. Select **Microsoft Access (ADO)**, and then click **Next**.



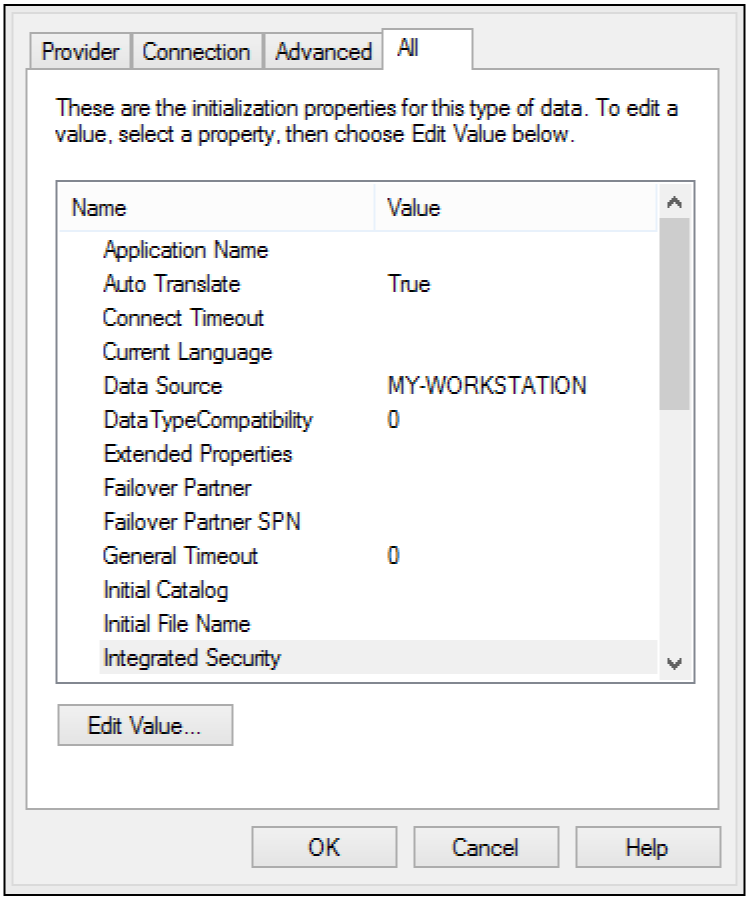
○ Use an existing MS Access database
● Create a new MS Access database
C:\Users\altova\Documents\Altova\products.mdb

3. Select **Create a new MS Access database** and then enter the path (either relative or absolute) of the DB file to create (for example, **c:\users\public\products.mdb**). Alternatively, click **Browse** to select a folder, type the name of the database file in the *Browse* text box (for example, **products.mdb**), and click **Save**. Make sure that you have write permissions to the folder where you want to create the database file and (ii) that the database file name must have the **.mdb** or **.accdb** extension.
4. Click **Connect**.

### 11.2.3.3 Set Up SQL Server Data Link Properties

If you connect to a Microsoft SQL Server database through ADO[967], check the following properties in the *All* tab of the Data Link Properties dialog box (*screenshot below*).

- *Integrated Security:* If SQL Server Native Client is the data provider on the Provider tab, set this property to a space character.
- *Persist Security Info:* Set this property to *True*.

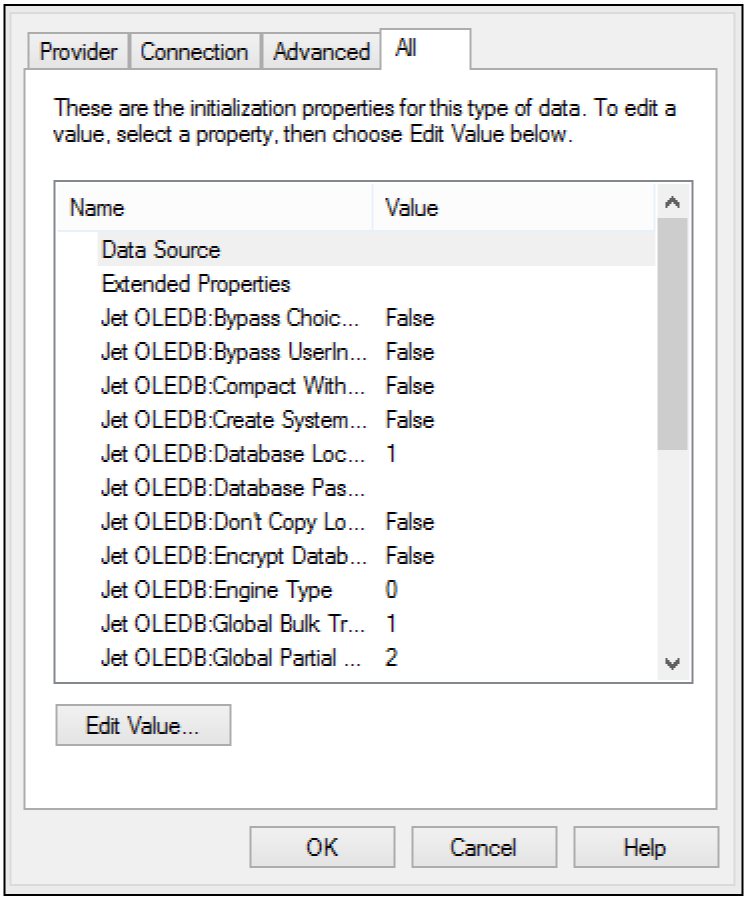


These are the initialization properties for this type of data. To edit a value, select a property, then choose Edit Value below.

| Name | Value |
|---|---|
| Application Name | |
| Auto Translate | True |
| Connect Timeout | |
| Current Language | |
| Data Source | MY-WORKSTATION |
| DataTypeCompatibility | 0 |
| Extended Properties | |
| Failover Partner | |
| Failover Partner SPN | |
| General Timeout | 0 |
| Initial Catalog | |
| Initial File Name | |
| Integrated Security | |

### 11.2.3.4 Set Up MS Access Data Link Properties

If you connect to a Microsoft Access database through ADO[967], then in the *All* tab of the Data Link Properties dialog box (*screenshot below*), check the properties listed below the screenshot.



These are the initialization properties for this type of data. To edit a value, select a property, then choose Edit Value below.

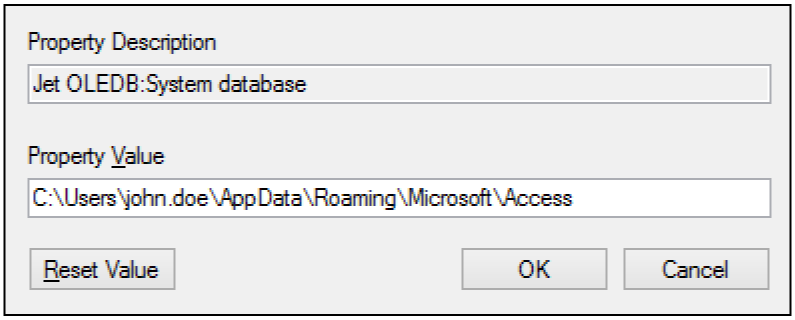
| Name | Value |
|---|---|
| Data Source | |
| Extended Properties | |
| Jet OLEDB:Bypass Choic... | False |
| Jet OLEDB:Bypass UserIn... | False |
| Jet OLEDB:Compact With... | False |
| Jet OLEDB:Create System... | False |
| Jet OLEDB:Database Loc... | 1 |
| Jet OLEDB:Database Pas... | |
| Jet OLEDB:Don't Copy Lo... | False |
| Jet OLEDB:Encrypt Datab... | False |
| Jet OLEDB:Engine Type | 0 |
| Jet OLEDB:Global Bulk Tr... | 1 |
| Jet OLEDB:Global Partial ... | 2 |

*__Data Source__*

This property stores the path to the Microsoft Access database file. We recommend using the UNC (Universal Naming Convention) path format, for example: `\\anyserver\share$\filepath`

*__Jet OLEDB:System Database__*

This property stores the path to the workgroup information file. You may need to explicitly set the value of this property before you can connect to a Microsoft Access database. If you cannot connect due to a "workgroup information file" error, locate the workgroup information file (`System.MDW`) applicable to your user profile, and set the property value to the path of the `System.MDW` file.



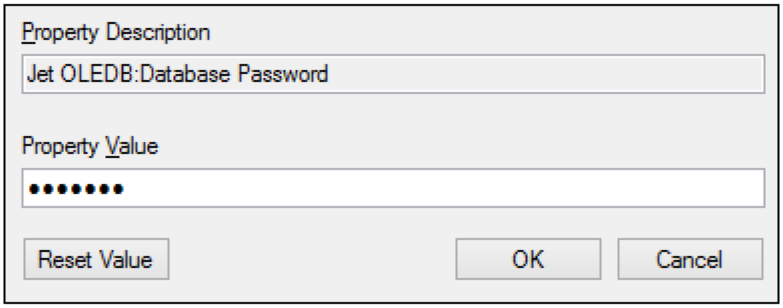
Property Description

Jet OLEDB:System database

Property Value

C:\Users\john.doe\AppData\Roaming\Microsoft\Access

*Jet OLEDB:Database Password*
If the database is password-protected, set the value of this property to the database password.


Property Description
Jet OLEDB:Database Password
Property Value
•••••••

## 11.2.4 ADO.NET Connection

ADO.NET is a set of Microsoft .NET Framework libraries designed to interact with data, including data from databases. To connect to a database from MobileTogether Designer through ADO.NET, Microsoft .NET Framework 4 or later is required. Connect to a database through ADO.NET by selecting a .NET provider and supplying a connection string.

A .NET data provider is a collection of classes that enables connecting to a particular type of data source (for example, a SQL Server, or an Oracle database), executing commands against it, and fetching data from it. So, when you use ADO.NET, MobileTogether Designer interacts with a database through a data provider—one that is optimized to work with the specific type of data source that it is designed for.

There are two types of .NET providers:

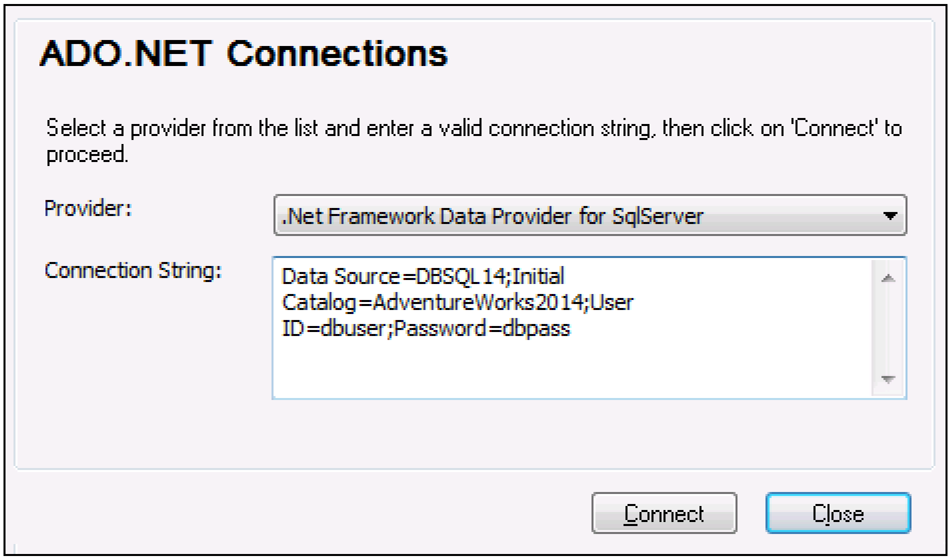
- Supplied by default with Microsoft .NET Framework.
- Supplied by major database vendors, as an extension to the .NET Framework. Such ADO.NET providers must be installed separately and can typically be downloaded from the website of the respective database vendor.

**Note:** Certain ADO.NET providers are not supported or have limited support. See ADO.NET Support Notes 978.

Set up the connection as follows.

1. Start the database connection wizard 961.
2. Click **ADO.NET Connections**.
3. Select a .NET data provider from the list. The list of providers available by default with the .NET Framework appears in the "Provider" list. Vendor-specific .NET data providers are available in the list only if they are already installed on your system. To become available, vendor-specific .NET providers must be installed into the GAC (Global Assembly Cache) by running the **`.msi`** or **`.exe`** file supplied by the database vendor.
4. Enter a database connection string. A connection string defines the database connection information as semicolon-delimited key/value pairs of connection parameters. For example, a connection string such as `Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass` connects to the SQL Server database **`ProductsDB`** on server **`DBSQLSERV`**, with the user name **`dbuser`** and password **`dbpass`**. You can create a connection string by

typing the key/value pairs directly into the "Connection String" dialog box. Another option is to create it with Visual Studio (see Creating a Connection String in Visual Studio 974). The syntax of the connection string depends on the provider selected from the "Provider" list. For examples, see Sample ADO.NET Connection Strings 977.



ADO.NET Connections

Select a provider from the list and enter a valid connection string, then click on 'Connect' to proceed.

Provider: .Net Framework Data Provider for SqlServer

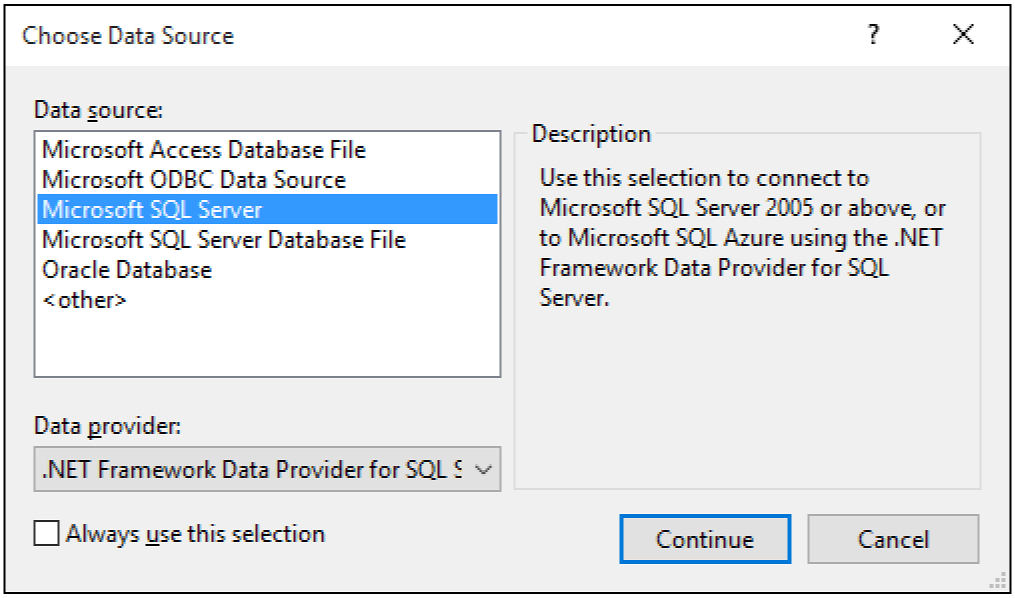
Connection String: Data Source=DBSQL14;Initial Catalog=AdventureWorks2014;User ID=dbuser;Password=dbpass

5. Click Connect.

### 11.2.4.1 Creating a Connection String in Visual Studio

In order to connect to a data source using ADO.NET, a valid database connection string is required. The following instructions show you how to create a connection string from Visual Studio.

1. On the **Tools** menu, click **Connect to Database**.
2. Select a data source from the list (in this example, Microsoft SQL Server). The Data Provider is filled automatically based on your choice.

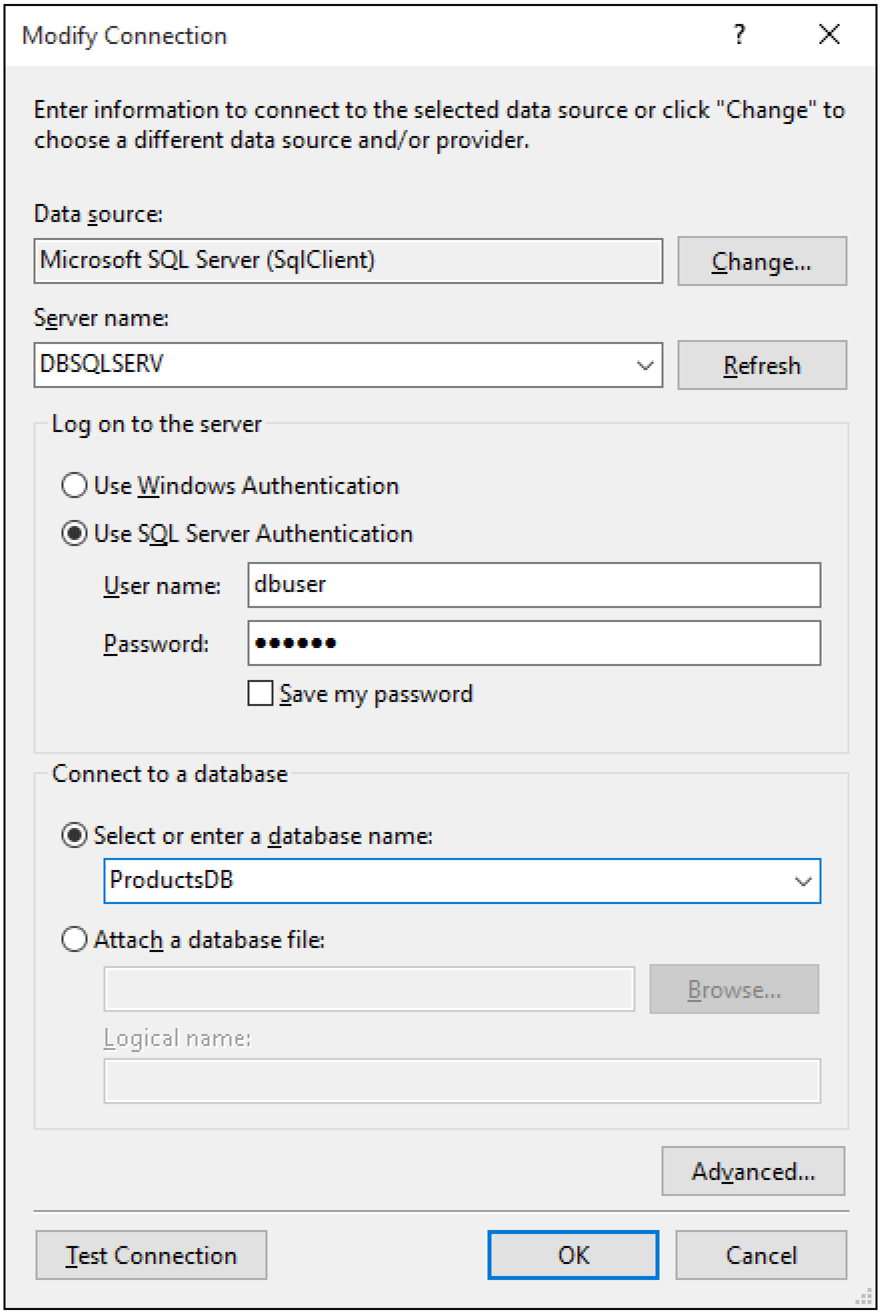

Choose Data Source
?
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server Database File
Oracle Database
<other>
Description
□ Always use this selection
Continue
Cancel

Data source: Microsoft SQL Server

Data provider: .NET Framework Data Provider for SQL S

Use this selection to connect to Microsoft SQL Server 2005 or above, or to Microsoft SQL Azure using the .NET Framework Data Provider for SQL Server.

3. Click **Continue**.



Modify Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: DBSQLSERV

Log on to the server

○ Use Windows Authentication

● Use SQL Server Authentication

User name: dbuser

Password: ●●●●●●

□ Save my password

Connect to a database

● Select or enter a database name: ProductsDB

○ Attach a database file:

Logical name:

4. Enter the server host name and the user name and password to the database. In this example, we are connecting to the database `ProductsDB` on server `DBSQLSERV`, using SQL Server authentication.
5. Click **OK**.

If the database connection is successful, it appears in the Server Explorer window. You can display the Server Explorer window using the menu command **View | Server Explorer**. To obtain the database connection string, right-click the connection in the Server Explorer window, and select **Properties**. The connection string is now displayed in the Properties window of Visual Studio. Note that, before pasting the string into the "Connection String" box of MobileTogether Designer, you will need to replace the asterisk ( * ) characters with the actual password.

## 11.2.4.2 Sample ADO.NET Connection Strings

To set up an ADO.NET connection, you need to select an ADO.NET provider from the database connection dialog box and enter a connection string (see also Setting up an ADO.NET Connection 973). Sample ADO.NET connection strings for various databases are listed below under the .NET provider where they apply.

*.NET Data Provider for Teradata*

This provider can be downloaded from Teradata website (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata). A sample connection string looks as follows:

```
Data Source=ServerAddress;User Id=user;Password=password;
```

*.NET Framework Data Provider for IBM i*

This provider is installed as part of *IBM i Access Client Solutions - Windows Application Package*. A sample connection string looks as follows:

```
DataSource=ServerAddress;UserID=user;Password=password;DataCompression=True;
```

For more information, see the ".NET Provider Technical Reference" help file included in the installation package above.

*.NET Framework Data Provider for MySQL*

This provider can be downloaded from MySQL website (https://dev.mysql.com/downloads/connector/net/). A sample connection string looks as follows:

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```

See also: https://dev.mysql.com/doc/connector-net/en/connector-net-programming-connecting-connection-string.html

*.NET Framework Data Provider for SQL Server*

A sample connection string looks as follows:

```
Data Source=DBSQLSERV;Initial Catalog=ProductsDB;User ID=dbuser;Password=dbpass
```

See also: https://msdn.microsoft.com/en-us/library/ms254500(v=vs.110).aspx

*IBM DB2 Data Provider 10.1.2 for .NET Framework 4.0*

A sample connection string looks as follows:

```
Database=PRODUCTS;UID=user;Password=password;Server=localhost:50000;
```

**Note:** This provider is typically installed with the IBM DB2 Data Server Client package. If the provider is missing from the list of ADO.NET providers after installing IBM DB2 Data Server Client package, refer to the following technical note: https://www-01.ibm.com/support/docview.wss?uid=swg21429586. See also: https://www.ibm.com/support/knowledgecenter/en/SSEPGG_10.1.0/com.ibm.swg.im.dbclient.adonet.ref.doc/doc/DB2ConnectionClassConnectionStringProperty.html

*Oracle Data Provider for .NET (ODP.NET)*

The installation package which includes the ODP.NET provider can be downloaded from the Oracle website (see http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html). A sample connection string looks as follows:

```
Data Source=DSORCL;User Id=user;Password=password;
```

where `DSORCL` is the name of the data source which points to an Oracle service name defined in the **`tnsnames.ora`** file, as described in Connecting to Oracle (ODBC) 1013.

To connect without configuring a service name in the **`tnsnames.ora`** file, use a string such as:

```
Data Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host)(PORT=port)))
(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME=MyOracleSID)));User
Id=user;Password=password;
```

See also: https://docs.oracle.com/cd/B28359_01/win.111/b28375/featConnecting.htm.

## 11.2.4.3 ADO.NET Support Notes

The following table lists known ADO.NET database drivers that are currently not supported or have limited support in MobileTogether Designer.

| Database | Driver | Support notes |
|---|---|---|
| All databases | **.Net Framework Data Provider for ODBC** | Limited support. Known issues exist with Microsoft Access connections. It is recommended to use ODBC direct connections instead. |
| | **.Net Framework Data Provider for OleDb** | Limited support. Known issues exist with Microsoft Access connections. It is recommended to use ADO direct connections instead. |
| Firebird | **Firebird ADO.NET Data Provider** | Limited support. It is recommended to use ODBC or JDBC instead. |
| Informix | **IBM Informix Data Provider for .NET Framework 4.0** | Not supported. Use **DB2 Data Server Provider** instead. |
| IBM DB2 for i (iSeries) | **.Net Framework Data Provider for i5/OS** | Not supported. Use **.Net Framework Data Provider for IBM i** instead, installed as part of the *IBM i Access Client Solutions - Windows Application Package*. |
| Oracle | **.Net Framework Data Provider for Oracle** | Limited support. Although this driver is provided with the .NET Framework, its usage is discouraged by Microsoft, because it is deprecated. |
| PostgreSQL | **—** | No ADO.NET drivers for this vendor are supported. Use a native connection instead. |

| Database | Driver | Support notes |
|---|---|---|
| Sybase | — | No ADO.NET drivers for this vendor are supported. |

## 11.2.5 JDBC Connection

JDBC (Java Database Connectivity) is a database access interface which is part of the Java software platform from Oracle. JDBC connections are generally more resource-intensive than ODBC connections but may provide features not available through ODBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC drivers from the database vendor must be installed. These may be JDBC drivers installed as part of a database client installation, or supported JDBC libraries (`.jar` files) that are downloaded separately. See also Database Connection Examples 989.
- The `CLASSPATH` environment variable must include the path to the JDBC driver (one or several `.jar` files) on your Windows operating system. When you install some database clients, the installer may configure this variable automatically. See also Configuring the CLASSPATH 981.

### Connect to SQL Server via JDBC with Windows credentials

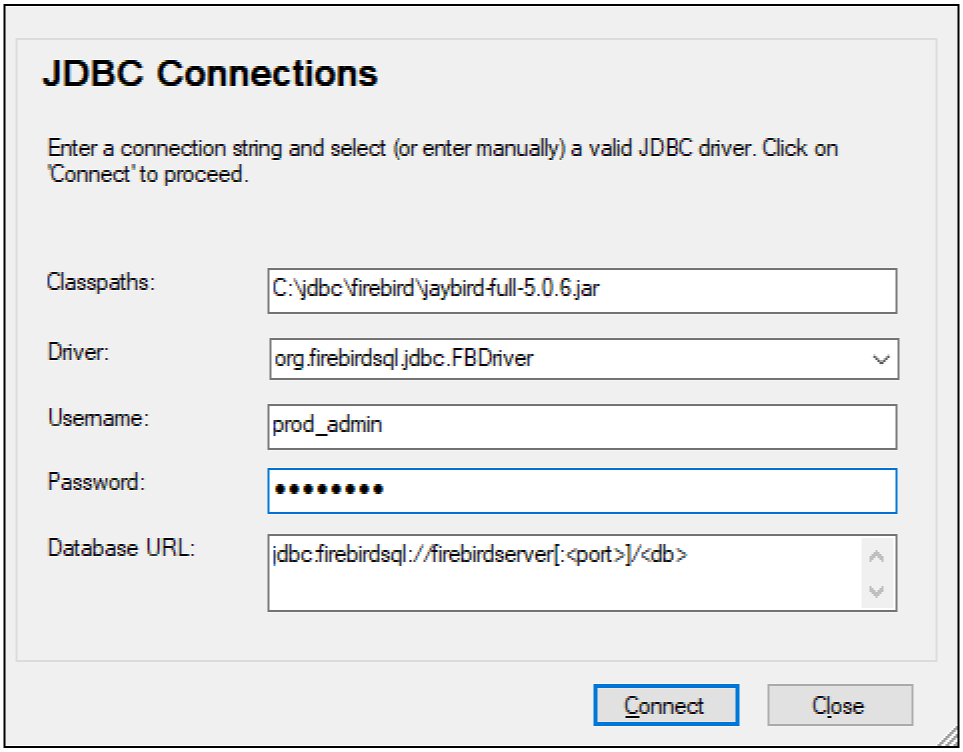
If you connect to SQL Server through JDBC with Windows credentials (integrated security), note the following:

- The **`sqljdbc_auth.dll`** file included in the JDBC driver package must be copied to a directory that is on the system `PATH` environment variable. There are two such files, one for the x86 and one for x64 platform. Make sure that you add to the `PATH` the one that corresponds to your JDK platform.
- The JDBC connection string must include the property `integratedSecurity=true`.

For further information, refer to *Microsoft JDBC driver for SQL Server* documentation, https://docs.microsoft.com/en-us/sql/connect/jdbc/building-the-connection-url.

### Set up a JDBC connection

1. Start the database connection wizard 961 and click JDBC Connections.
2. If required, enter a semicolon-separated list of `.jar` file paths in the *Classpaths* field. The `.jar` libraries entered here will be loaded into the environment in addition to those already defined in the `CLASSPATH` 981 environment variable. JDBC drivers found in the source .jar libraries referenced via the *Classpaths* field and the system's `CLASSPATH` 981 are listed in the Driver dropdown list (see next step).



JDBC Connections

Enter a connection string and select (or enter manually) a valid JDBC driver. Click on 'Connect' to proceed.

Classpaths: C:\jdbc\firebird\jaybird-full-5.0.6.jar

Driver: org.firebirdsql.jdbc.FBDriver

Username: prod_admin

Password: ••••••••

Database URL: jdbc:firebirdsql://firebirdserver[:<port>]/<db>

3. In the *Driver* field, select a JDBC driver from the list or enter a Java class name. The list will contain the JDBC drivers configured through in the *Classpaths* field (see above) and the CLASSPATH 981 environment variable.

> The JDBC driver paths defined in the CLASSPATH variable, as well as any .jar file paths entered directly in the database connection dialog box are all supplied to the Java Virtual Machine (JVM). The JVM then decides which drivers to use in order to establish a connection. It is recommended that you keep track of Java classes loaded into the JVM so as not to create potential JDBC driver conflicts and avoid unexpected results when connecting to the database.

4. Enter the username and password of the database in the corresponding fields.
5. In the *Database URL* field, enter the JDBC connection URL (JDBC connection string) in the format specific to your database type. The following table describes the syntax of JDBC connection strings for common database types.

| Database | JDBC Connection URL |
|---|---|
| Firebird | `jdbc:firebirdsql://<host>[:<port>]/<database path or alias>` |
| IBM DB2 | `jdbc:db2://hostName:port/databaseName` |
| IBM DB2 for i | `jdbc:as400://[host]` |
| IBM Informix | `jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver` |

| Database | JDBC Connection URL |
|---|---|
| MariaDB | `jdbc:mariadb://hostName:port/databaseName` |
| Microsoft SQL Server | `jdbc:sqlserver://hostName:port;databaseName=name` |
| MySQL | `jdbc:mysql://hostName:port/databaseName` |
| Oracle | `jdbc:oracle:thin:@hostName:port:SID`<br>`jdbc:oracle:thin:@//hostName:port/service` |
| Oracle XML DB | `jdbc:oracle:oci:@//hostName:port:service` |
| PostgreSQL | `jdbc:postgresql://hostName:port/databaseName` |
| Progress OpenEdge | `jdbc:datadirect:openedge://host:port;databaseName=db_name` |
| Sybase | `jdbc:sybase:Tds:hostName:port/databaseName` |
| Teradata | `jdbc:teradata://databaseServerName` |

Note that syntax variations of the formats listed above are also possible. For example, the database URL may exclude the port or may include the username and password of the database. Check the documentation of the database vendor for further details.

6. Click **Connect**.

## 11.2.5.1 Configuring the CLASSPATH

The `CLASSPATH` environment variable is used by the Java Runtime Environment (JRE) or the Java Development Kit (JDK) to locate Java classes and other resource files on your operating system. When you connect to a database through JDBC, this variable must be configured to include the path to the JDBC driver on your operating system, and, in some cases, the path to additional library files specific to the database type you are using.

The following table lists sample file paths that must be typically included in the `CLASSPATH` variable. Importantly, you may need to adjust this information based on the location of the JDBC driver on your system, the JDBC driver name, as well as the JRE/JDK version present on your operating system. To avoid connectivity problems, check the installation instructions and any pre-installation or post-installation configuration steps applicable to the JDBC driver installed on your operating system.

| Database | Sample CLASSPATH entries |
|---|---|
| Firebird | `C:\Program Files\Firebird\Jaybird-2.2.8-JDK_1.8\jaybird-full-2.2.8.jar` |
| IBM DB2 | `C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;` |
| IBM DB2 for i | `C:\jt400\jt400.jar;` |
| IBM Informix | `C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;` |

| Database | Sample CLASSPATH entries |
|---|---|
| Microsoft SQL Server | `C:\Program Files\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar` |
| MariaDB | `<installation directory>\mariadb-java-client-2.2.0.jar` |
| MySQL | `<installation directory>\mysql-connector-java-`*version*`-bin.jar;` |
| Oracle | `ORACLE_HOME\jdbc\lib\ojdbc6.jar;` |
| Oracle (with XML DB) | `ORACLE_HOME\jdbc\lib\ojdbc6.jar;ORACLE_HOME\LIB\xmlparserv2.jar;ORACLE_HOME\RDBMS\jlib\xdb.jar;` |
| PostgreSQL | `<installation directory>\postgresql.jar` |
| Progress OpenEdge | `%DLC%\java\openedge.jar;%DLC%\java\pool.jar;`<br>Note: Assuming the Progress OpenEdge SDK is installed on the machine, `%DLC%` is the directory where OpenEdge is installed. |
| Sybase | `C:\sybase\jConnect-7_0\classes\jconn4.jar` |
| Teradata | `<installation directory>\tdgssconfig.jar;<installation directory>\terajdbc4.jar` |

Note the following points:

- The `CLASSPATH` setting is available in the Environment Variables setting of your Windows system. Include in the `CLASSPATH` the path where the JDBC driver is located, separating it from other paths by a semi-colon.
- Changing the `CLASSPATH` variable may affect the behavior of Java applications on your machine. To understand possible implications before you proceed, refer to the Java documentation.
- Environment variables can be user or system. To change system environment variables, you need administrative rights on the operating system.
- After you change the environment variable, restart any running programs for settings to take effect. Alternatively, log off or restart your operating system.

## 11.2.6 ODBC Connection

ODBC (Open Database Connectivity) is a widely used data access technology that enables you to connect to a database from MobileTogether Designer. It can be used either as primary means to connect to a database, or as an alternative to native, OLE DB, or JDBC-driven connections.

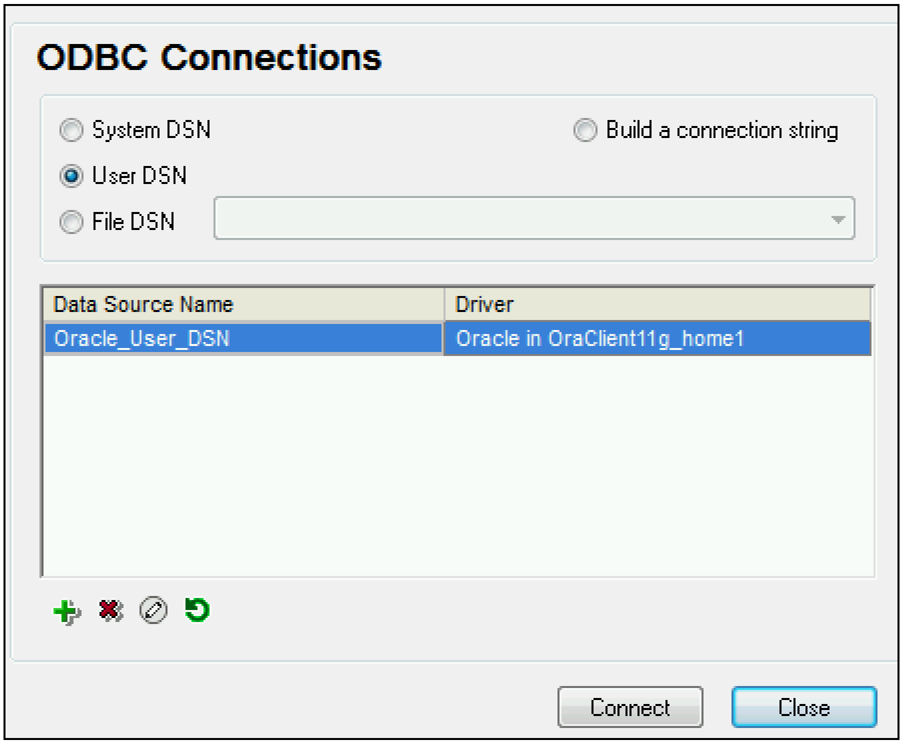
To connect to a database through ODBC, first you need to create an ODBC data source name (DSN) on the operating system. This step is not required if the DSN has already been created, perhaps by another user of the operating system. The DSN represents a uniform way to describe the database connection to any ODBC-aware client application on the operating system, including MobileTogether Designer. DSNs can be of the following types:

- System DSN

- User DSN
- File DSN

A *System* data source is accessible by all users with privileges on the operating system. A *User* data source is available to the user who created it. Finally, if you create a *File DSN*, the data source will be created as a file with the .dsn extension which you can share with other users, provided that they have installed the drivers used by the data source.

Any DSNs already available on your machine are listed by the database connection dialog box when you click **ODBC connections** on the ODBC connections dialog box (*screenshot below*).



ODBC Connections

○ System DSN ○ Build a connection string

● User DSN

○ File DSN

| Data Source Name | Driver |
|---|---|
| Oracle_User_DSN | Oracle in OraClient11g_home1 |

If a DSN to the required database is not available, the MobileTogether Designer database connection wizard will assist you to create it; however, you can also create it directly on your Windows operating system. In either case, before you proceed, ensure that the ODBC driver applicable for your database is in the list of ODBC drivers available to the operating system (see Viewing the Available ODBC Drivers 984 ).

## Connect with a new DSN

To connect with a new DSN, carry out the following steps. Note that, to create a System DSN, you need administrative rights on the operating system and MobileTogether Designer must be run as administrator..

1. Start the database connection wizard 961 .
2. On the database connection dialog box, click **ODBC Connections**.
3. Select a data source type (User DSN, System DSN, File DSN).
4. Click **Add** ✚.
5. Select a driver, and then click **User DSN** or **System DSN** (depending on the type of the DSN you want to create). If the driver applicable to your database is not listed, download it from the database vendor and install it (see Database Drivers Overview 963 ).

6. On the dialog box that pops up, fill in any driver specific connection information to complete the setup.

For the connection to be successful, you will need to provide the host name (or IP address) of the database server, as well as the database username and password. There may be other optional connection parameters—these parameters vary between database providers. For detailed information about the parameters specific to each connection method, consult the documentation of the driver provider. Once created, the DSN becomes available in the list of data source names. This enables you to reuse the database connection details any time you want to connect to the database. Note that User DSNs are added to the list of User DSNs whereas System DSNs are added to the list of System DSNs.

### Connect with an existing DSN

To connect with an existing DSN, do the following.

1. Start the database connection wizard 961.
2. Click **ODBC Connections**.
3. Choose the type of the existing data source (User DSN, System DSN, File DSN).
4. Click the existing DSN record, and then click **Connect**.

### Build a connection string based on an existing .dsn file

Do this as follows.

1. Start the database connection wizard 961.
2. Click **ODBC Connections**.
3. Select **Build a connection string** and then click **Build**.
4. If you want to build the connection string using a File DSN, click the *File Data Source* tab. Otherwise, click the *Machine Data Source* tab. (System DSNs and User DSNs are known as "Machine" data sources.)
5. Select the required **.dsn** file, and then click **OK**.

### Connect by using a prepared connection string

Do this as follows.

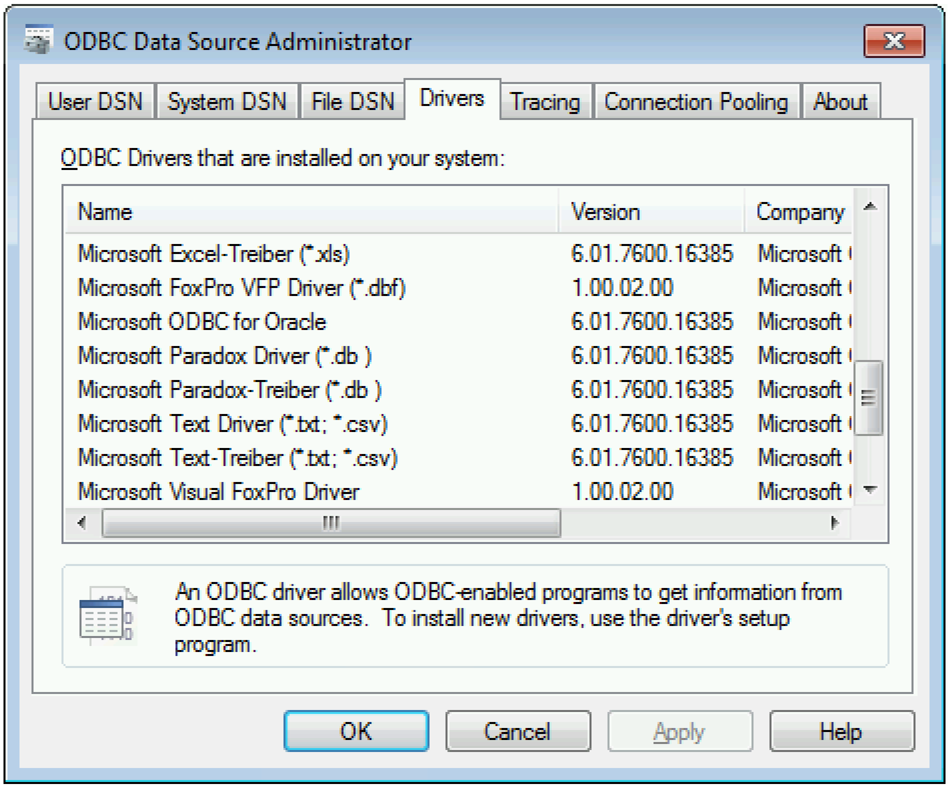
1. Start the database connection wizard 961.
2. Click **ODBC Connections**.
3. Select **Build a connection string**.
4. Paste the connection string into the provided box, and then click **Connect**.

## 11.2.6.1 Available ODBC Drivers

You can view the ODBC drivers available on your operating system in the ODBC Data Source Administrator. You can access the ODBC Data Source Administrator (**Odbcad32.exe**) from the Windows Control Panel, under Administrative Tools. On 64-bit operating systems, there are two versions of this executable:

- The 32-bit version of the **Odbcad32.exe** file is located in the **C:\Windows\SysWoW64** directory (assuming that **C:** is your system drive).
- The 64-bit version of the **Odbcad32.exe** file is located in the C**:\Windows\System32** directory.

Any installed 32-bit database drivers are visible in the 32-bit version of ODBC Data Source Administrator (*screenshot below*), while 64-bit drivers—in the 64-bit version. Therefore, ensure that you check the database drivers from the relevant version of ODBC Data Source Administrator.



ODBC Drivers that are installed on your system:

| Name | Version | Company |
|---|---|---|
| Microsoft Excel-Treiber (*.xls) | 6.01.7600.16385 | Microsoft ( |
| Microsoft FoxPro VFP Driver (*.dbf) | 1.00.02.00 | Microsoft ( |
| Microsoft ODBC for Oracle | 6.01.7600.16385 | Microsoft ( |
| Microsoft Paradox Driver (*.db ) | 6.01.7600.16385 | Microsoft ( |
| Microsoft Paradox-Treiber (*.db ) | 6.01.7600.16385 | Microsoft ( |
| Microsoft Text Driver (*.txt; *.csv) | 6.01.7600.16385 | Microsoft ( |
| Microsoft Text-Treiber (*.txt; *.csv) | 6.01.7600.16385 | Microsoft ( |
| Microsoft Visual FoxPro Driver | 1.00.02.00 | Microsoft ( |

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

If the driver to your target database does not exist in the list, or if you want to add an alternative driver, you will need to download it from the database vendor (see Database Drivers Overview 963 ). Once the ODBC driver is available on your system, you are ready to create ODBC connections with it (see Setting up an ODBC Connection 982).

## 11.2.7 SQLite Connection

SQLite is a file-based, self-contained database type, which makes it ideal in scenarios where portability and ease of configuration is important. Since SQLite databases are natively supported by MobileTogether Designer, you do not need to install any drivers to connect to them.

### SQLite database support notes

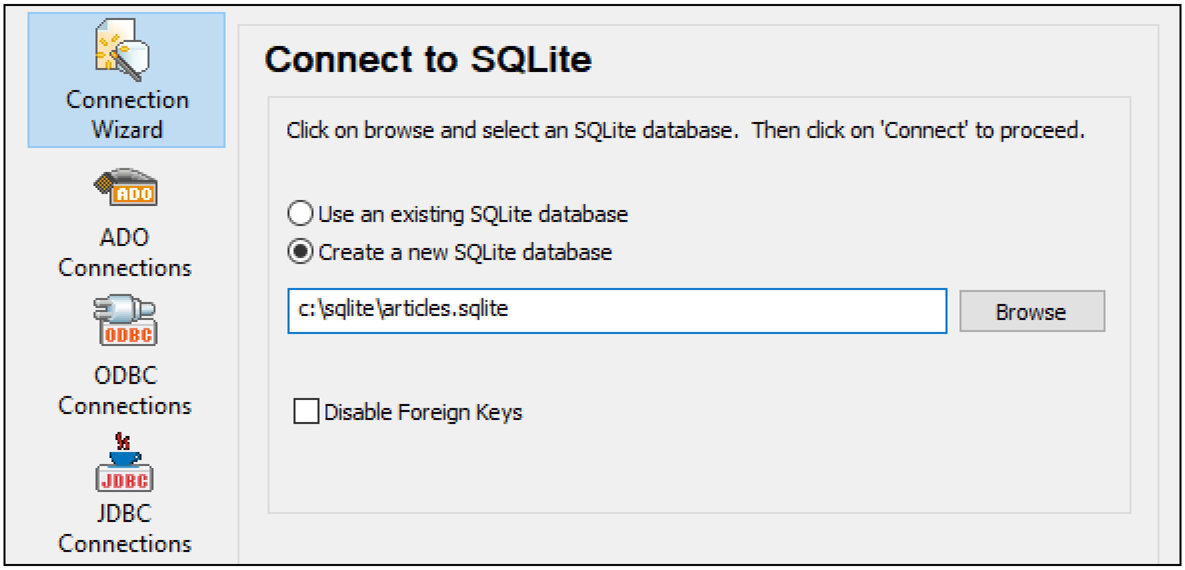
- On Linux, statement execution timeout for SQLite databases is not supported.
- Full text search tables are not supported.
- SQLite allows values of different data types in each row of a given table. All processed values must be compatible with the declared column type; therefore, unexpected values can be retrieved and run-time errors may occur if your SQLite database has row values which are not the same as the declared column type.

**Important:** It is recommended to create tables with the `STRICT` keyword to ensure more predictable behavior of your data. Otherwise, the data may not be read or written correctly when values of different types are mixed in one column. To find out more about STRICT tables, see the SQLite documentation.

### 11.2.7.1 Create a New SQLite Database

If you want to create a new SQLite database file and connect to it, follow the steps below. The database file created by MobileTogether Designer will be empty. Use queries or scripts to create the required database structure and fill in data.

1. Run the database connection wizard (see Starting the Database Connection Wizard[961]).
2. Select **SQLite**, and then click **Next**.



Connect to SQLite

Click on browse and select an SQLite database. Then click on 'Connect' to proceed.

○ Use an existing SQLite database
● Create a new SQLite database

c:\sqlite\articles.sqlite

□ Disable Foreign Keys

3. Select **Create a new SQLite database**, and then enter the path (either relative or absolute) of the database file to be created (for example, **`c:\users\public\products.sqlite`**). Alternatively, click **Browse** to select a folder, type the name of the database file in the *File name* text box, and click **Save**.

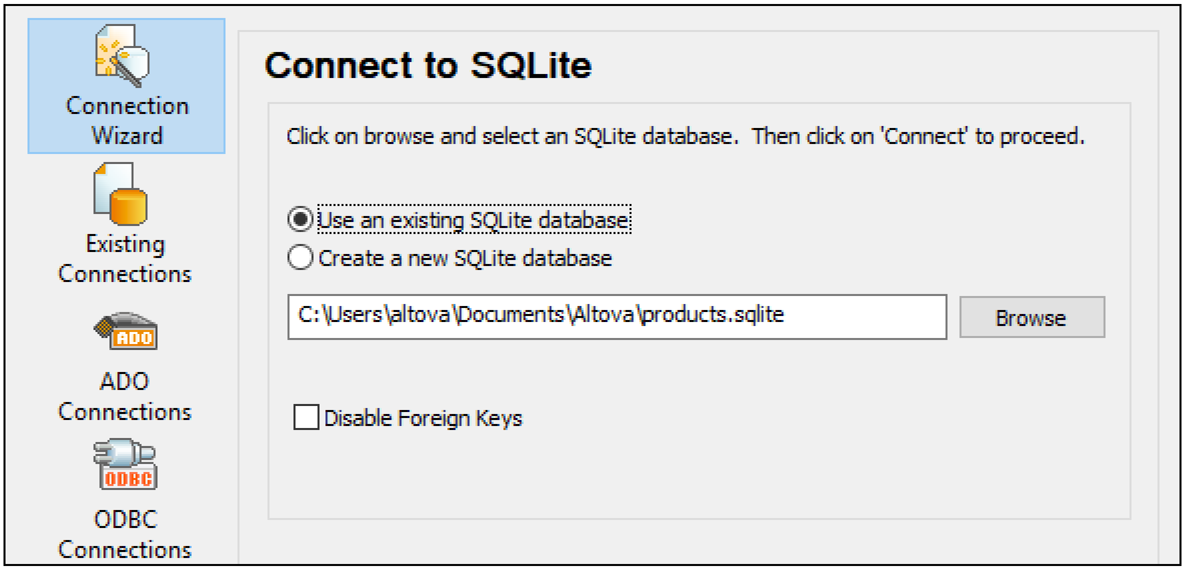
> Make sure that you have write permissions to the folder where you want to create the database file.

4. Optionally, select the **Disable Foreign Keys** check box, see Foreign Key Constraints[986].
5. Click **Connect**.

### 11.2.7.2 Foreign Key Constraints

When you connect to an existing SQLite database from MobileTogether Designer, or when you create a new one, foreign key constraints are enabled by default. Foreign key constraints help preserve the integrity of data in your database. For example, when foreign keys are enabled, it is not possible to delete a record from a table if it has dependencies in another table.

In certain cases, you may want to temporarily override this behavior and disable foreign keys, perhaps, in order to update or insert multiple rows of data without getting data validation errors. To explicitly disable foreign keys before connecting to the SQLite database, select the **Disable Foreign Keys** option available on the database connection wizard.



Connect to SQLite

Click on browse and select an SQLite database. Then click on 'Connect' to proceed.

● Use an existing SQLite database
○ Create a new SQLite database

C:\Users\altova\Documents\Altova\products.sqlite

□ Disable Foreign Keys

When foreign keys are disabled, you will be able to perform operations on data that would otherwise not be possible due to validation checks. At the same time, however, there is also the risk of introducing incorrect data into the database, or creating "orphaned" rows. (An example of an "orphaned" row would be an address in the "addresses" table not linked to any person in the "person" table, because the person was deleted but its associated address was not.)

## 11.2.8 Native Connections

Native connections are direct connections to a database's own network protocols and drivers. Therefore, no additional drivers need to be installed. Native connections provide the most efficient method of interacting with the database and full feature support.

If you intend to deploy files for execution on a Linux server, you do not need to install drivers on the target server.

You can set up native connections to the following DBs:

- MariaDB
- MySQL
- SQLite
- PostgreSQL

### Connection setup

To set up a native connection, follow the steps below:

1. Start the database connection wizard[961].
2. Select the DB you want to connect to.
3. In the dialog that appears, enter the relevant connection details, for example, the host (e.g., *localhost*), optionally the port (typically 5432), SSL Mode in the case of MySQL, the database name, username, and password in the corresponding boxes.
4. Click **Connect**.

### SQLite conections

For detailed information about SQLite connections, see the topic SQLite Connection[985].

### Notes for PostrgreSQL

If the PostgreSQL database server is on a different machine, note the following:

- The PostgreSQL database server must be configured to accept connections from clients. Specifically, the **pg_hba.conf** file must be configured to allow non-local connections. Secondly, the **postgresql.conf** file must be configured to listen on specified IP address(es) and port. For more information, check the PostgreSQL documentation (https://www.postgresql.org/docs/9.5/static/client-authentication-problems.html).
- The server machine must be configured to accept connections on the designated port (typically, 5432) through the firewall. For example, on a database server running on Windows, a rule may need to be created to allow connections on port 5432 through the firewall, from **Control Panel > Windows Firewall > Advanced Settings > Inbound Rules**.

## 11.2.9 Global Resources

After you have created a database as a global resource, its connection details are stored and can be used across all Altova products installed on your machine.

### Create a database as a global resource

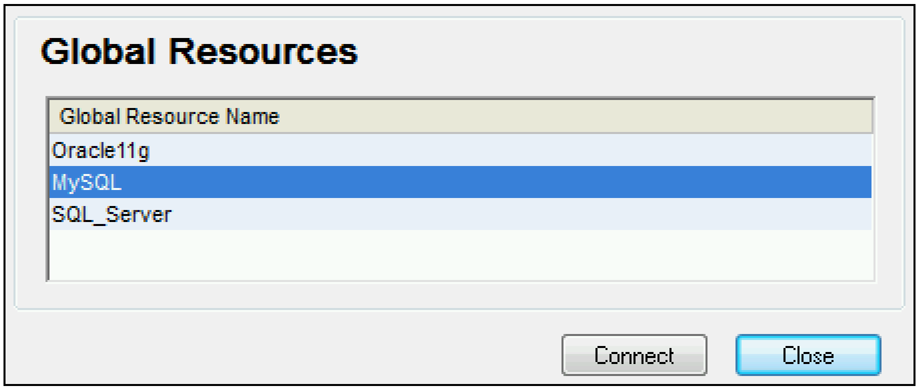
To create a database as a global resource, do the following

1. On the **Tools** menu of MobileTogether Designer, click **Global Resources**.
2. Click **Add**, and then click Database.
3. Type in a name for the global resource in the *Resource Alias* field.
4. Click **Choose Database**. The Connection Wizard[961] appears.
5. Use the Connection Wizard to add a database connection as described above.

### Use a global-resource database

To use a database that has been created as a global resource (*see above*), do the following:

1. Start the Connection Wizard as described above.
2. Select Global Resources. All the databases that have been created as global resources will be listed by their names in the Global Resources pane (*see screenshot below*).



3. Select the global resource that you want. Tip: Move the mouse cursor over a global resource in the list to see information about the database.

## 11.2.10 Database Connection Examples

This section includes examples for connecting to a database from MobileTogether Designer through ADO, ODBC, or JDBC. The ADO.NET connection examples are listed separately, see Sample ADO.NET Connection Strings 977. For instructions about establishing a native connection to PostgreSQL and SQLite, see Setting up a PostgreSQL Connection 987 and Setting up a SQLite Connection 985, respectively.

Note the following points:

- The instructions may differ if your Windows configuration, network environment and the database client or server software are not the same as the ones described in each example.
- For most database types, it is possible to connect using more than one data access technology (ADO, ADO.NET, ODBC, JDBC) or driver. The performance of the database connection, as well as its features and limitations will depend on the selected driver, database client software (if applicable), and any additional connectivity parameters that you may have configured outside MobileTogether Designer.

### 11.2.10.1 Firebird (JDBC)

This example illustrates how to connect to a Firebird database via JDBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The Firebird JDBC driver must be available on your operating system (it takes the form of a .jar file which provides connectivity to the database). The driver can be downloaded from the Firebird website ( https://www.firebirdsql.org/ ). This example uses *Jaybird 5.0.6*.

- You have the following database connection details: host, port, database path, name, or alias, username, and password.

### Connection

1. Start the database connection wizard 961 and click **JDBC Connections** (*see JDBC Connection 979 for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH 981 of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.
4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the Firebird example*

| Field | Value |
|---|---|
| Classpaths | `C:\jdbc\firebird\jaybird-full-5.0.6.jar` |
| Driver | *org.firebirdsql.jdbc.FBDriver* |
| Database URL | `jdbc:firebirdsql://<host>[:<port>]/<database path or alias>` |

## 11.2.10.2 Firebird (ODBC)

This example illustrates how to connect to a Firebird 2.5.4 database running on a Linux server.

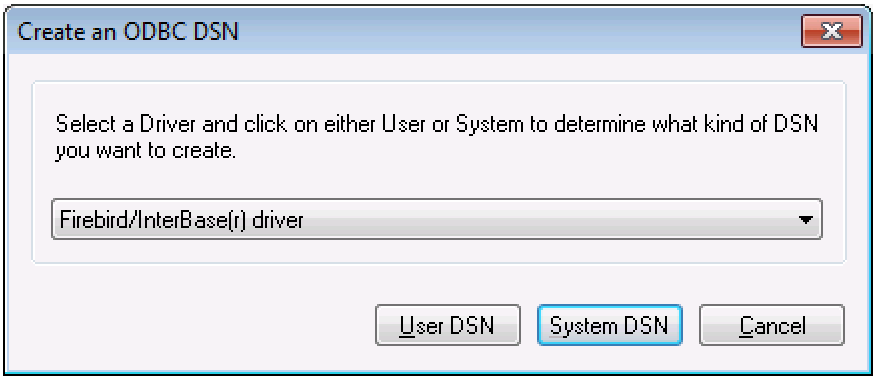
### Prerequisites

- The Firebird database server is configured to accept TCP/IP connections from clients.
- The Firebird ODBC driver must be installed on your operating system. This example uses the Firebird ODBC driver version 2.0.3.154 downloaded from the Firebird website.
- The Firebird client must be installed on your operating system. Note that there is no standalone installer available for the Firebird 2.5.4 client; the client is part of the Firebird server installation package. You can download the Firebird server installation package from the Firebird website. Look for *Windows executable installer for full Superclassic/Classic or Superserver*. To install only the client files, choose *Minimum client install - no server, no tools* when going through the wizard steps.
- You have the following database connection details: server host name or IP address, database path (or alias) on the server, user name, and password.

**Important:** The platform of both the Firebird ODBC driver and client (32-bit or 64-bit) must correspond to that of MobileTogether Designer. Additionally, the version of the Firebird client must correspond to the version of Firebird server to which you are connecting.

## Connection

1. Start the database connection wizard 961 and click **ODBC Connections**.
2. Select *User DSN* (or *System DSN* if you have administrative privileges), and then click **Create a New DSN** .

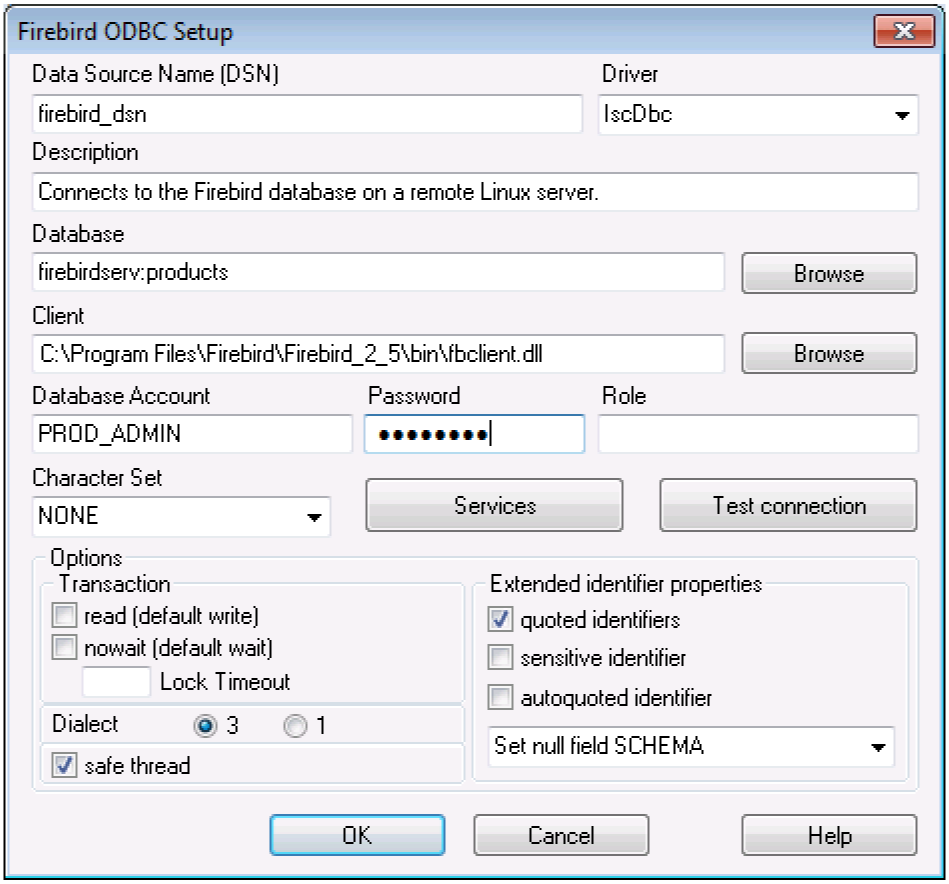


Create an ODBC DSN

Select a Driver and click on either User or System to determine what kind of DSN you want to create.

Firebird/InterBase(r) driver

3. Select the Firebird driver, and then click *User DSN* or *System DSN*, depending on what you selected in the previous step. If the Firebird driver is not available in the list, make sure that it is installed on your operating system (see also Viewing the Available ODBC Drivers 984).



Firebird ODBC Setup

Data Source Name (DSN): firebird_dsn

Driver: IscDbc

Description: Connects to the Firebird database on a remote Linux server.

Database: firebirdserv:products

Client: C:\Program Files\Firebird\Firebird_2_5\bin\fbclient.dll

Database Account: PROD_ADMIN

Password: ••••••••

Role:

Character Set: NONE

Options

Transaction

□ read (default write)

□ nowait (default wait)

Lock Timeout

Dialect ● 3 ○ 1

☑ safe thread

Extended identifier properties

☑ quoted identifiers

□ sensitive identifier

□ autoquoted identifier

Set null field SCHEMA

4. Enter the database connection details as follows:

| | |
|---|---|
| *Data Source Name (DSN)* | Enter a descriptive name for the data source you are creating. |
| *Database* | Enter the server host name or IP address, followed by a colon, followed by the database alias (or path). In this example, the host name is `firebirdserv`, and the database alias is `products`, as follows:<br>`firebirdserv:products`<br>Using a database alias assumes that, on the server side, the database administrator has configured the alias *products* to point to the actual Firebird (.fdb) database file on the server (see the Firebird documentation for more details).<br>You can also use the server IP address instead of the host name, and a path instead of an alias; therefore, any of the following sample connection strings are valid:<br>`firebirdserver:/var/Firebird/databases/butterflies.fdb`<br>`127.0.0.1:D:\Misc\Lenders.fdb`<br>If the database is on the local Windows machine, click **Browse** and select the Firebird (.fdb) database file directly. |
| *Client* | Enter the path to the **fbclient.dll** file. By default, this is the `bin` subdirectory of the Firebird installation directory. |
| *Database Account* | Enter the database user name supplied by the database administrator (in this example, `PROD_ADMIN`). |
| *Password* | Enter the database password supplied by the database administrator. |

5. Click **OK** to finish.

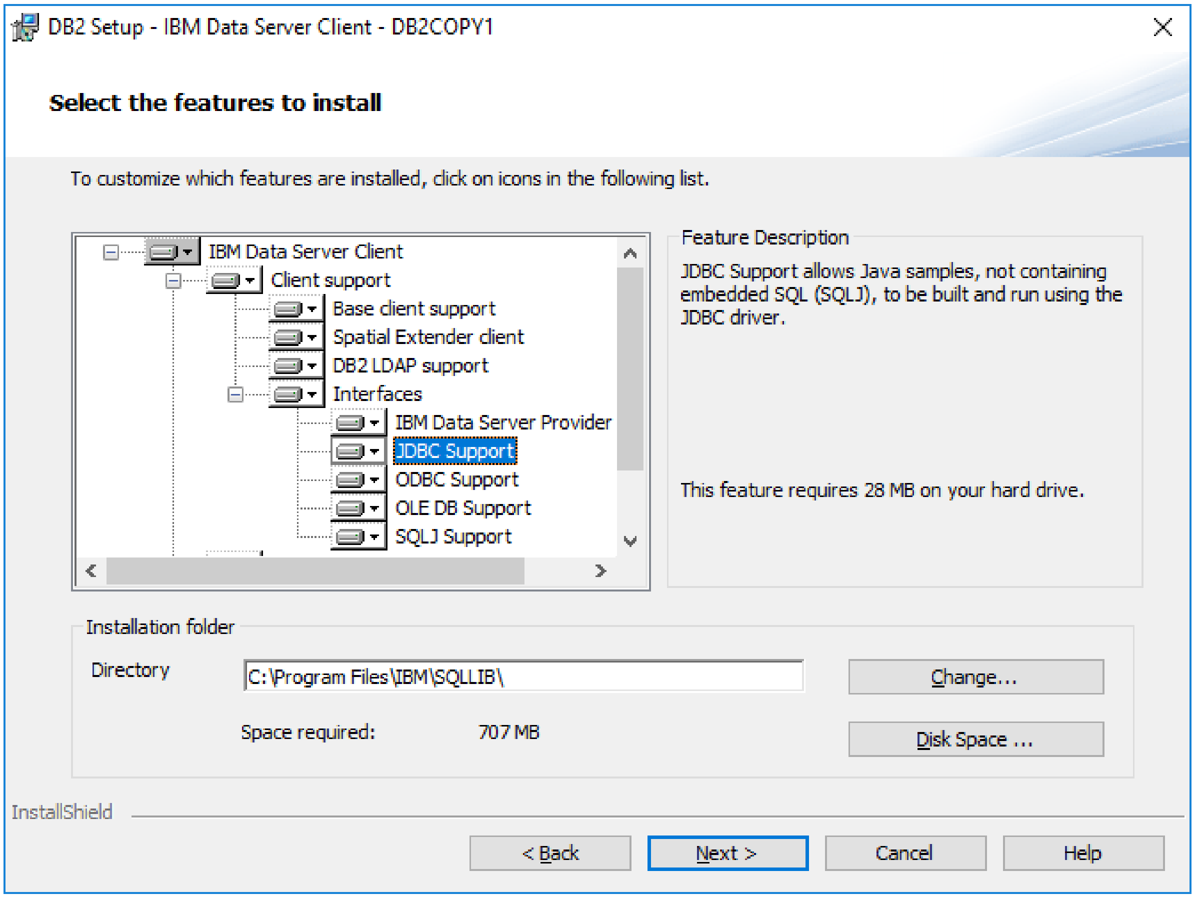
## 11.2.10.3 IBM DB2 (JDBC)

This example illustrates how to connect to an IBM DB2 database server via JDBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC driver (one or several .jar files that provide connectivity to the database) must be available on your operating system. This example uses the JDBC driver available after installing the IBM Data

Server Client version 10.1 (64-bit). For the JDBC drivers to be installed, choose a *Typical* installation, or select this option explicitly on the installation wizard.



DB2 Setup - IBM Data Server Client - DB2COPY1

Select the features to install

To customize which features are installed, click on icons in the following list.

Feature Description

JDBC Support allows Java samples, not containing embedded SQL (SQLJ), to be built and run using the JDBC driver.

This feature requires 28 MB on your hard drive.

Installation folder

Directory C:\Program Files\IBM\SQLLIB\

Space required: 707 MB

If you did not change the default installation path, the required `.jar` files will be in the `C:\Program Files\IBM\SQLLIB\java` directory after installation.

- You have the following database connection details: host, port, database path, name, or alias, username, and password.

## Connection

1. Start the database connection wizard 961 and click **JDBC Connections** (*see* *JDBC Connection* 979 *for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH 981 of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.
4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the IBM DB2 example*

| Field | Value |
|---|---|
| Classpaths | `C:\Program Files\IBM\SQLLIB\java\db2jcc.jar` |
| Driver | *com.ibm.db2.jcc.DB2Driver* |
| Database URL | `jdbc:db2://hostName:port/databaseName` |

## 11.2.10.4 IBM DB2 (ODBC)

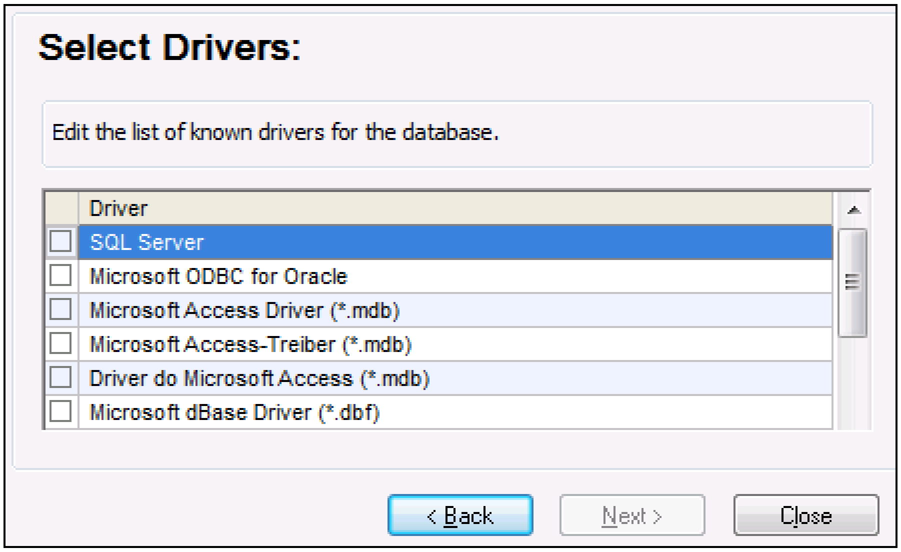
This example illustrates how to connect to an IBM DB2 database via ODBC.

### Prerequisites

- IBM Data Server Client must be installed and configured on your operating system (this example uses IBM Data Server Client 9.7). For installation instructions, check the documentation supplied with your IBM DB2 software. After installing the IBM Data Server Client, check if the ODBC drivers are available on your machine (see Viewing the Available ODBC Drivers 984).
- Create a database alias. There are several ways to do this:
    - From IBM DB2 Configuration Assistant
    - From IBM DB2 Command Line Processor
    - From the ODBC data source wizard (for this case, the instructions are shown below)
- You have the following database connection details: host, database, port, username, and password.

### Connection

1. Start the database connection wizard 961, select **Connection Wizard** and then *IBM DB2 (ODBC/JDBC)*. Click **Next**.
2. Select *ODBC* and click **Next**.
3. If prompted to edit the list of known drivers for the database, select the database drivers applicable to IBM DB2 (see *Prerequisites* above) and click **Next**.

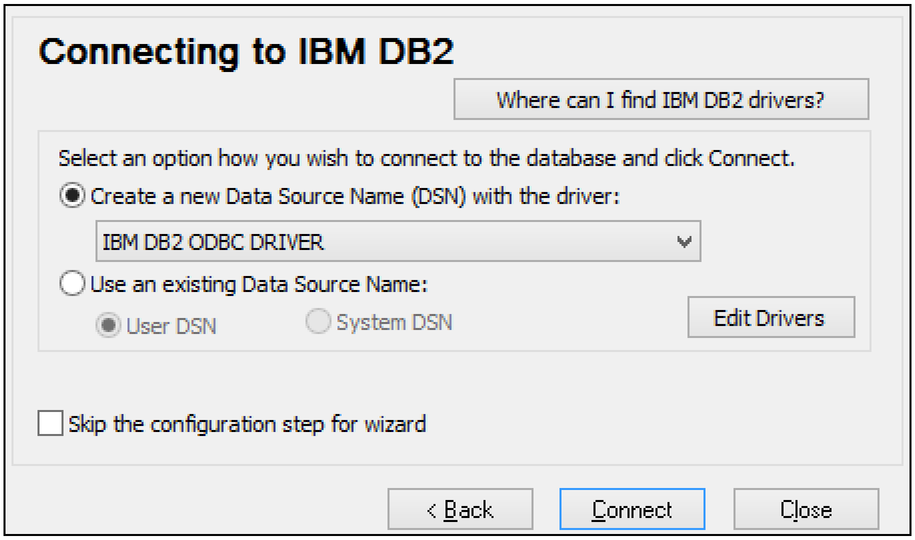


**Select Drivers:**

Edit the list of known drivers for the database.

| | Driver |
|---|---|
| ☐ | SQL Server |
| ☐ | Microsoft ODBC for Oracle |
| ☐ | Microsoft Access Driver (*.mdb) |
| ☐ | Microsoft Access-Treiber (*.mdb) |
| ☐ | Driver do Microsoft Access (*.mdb) |
| ☐ | Microsoft dBase Driver (*.dbf) |

4. Select the IBM DB2 driver you want from the list, and then click **Connect**. (To edit the list of available drivers, click **Edit Drivers**, and then check or uncheck the IBM DB2 drivers you wish to add or remove, respectively.)



**Connecting to IBM DB2**

Select an option how you wish to connect to the database and click Connect.

● Create a new Data Source Name (DSN) with the driver:

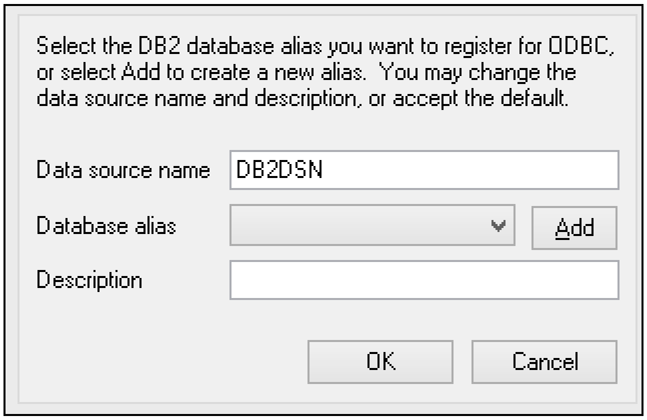
IBM DB2 ODBC DRIVER

○ Use an existing Data Source Name:

● User DSN ○ System DSN

☐ Skip the configuration step for wizard

5. Enter a data source name (in the screenshot below, DB2DSN), and then click **Add**.



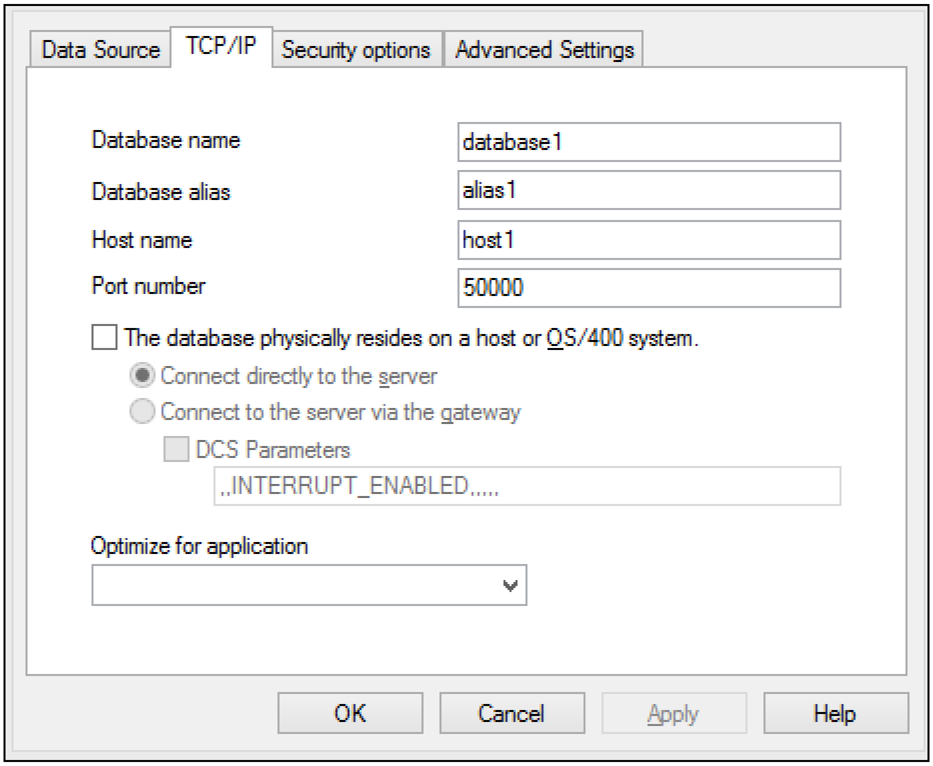
Select the DB2 database alias you want to register for ODBC, or select Add to create a new alias. You may change the data source name and description, or accept the default.

Data source name: DB2DSN

Database alias:

Description:

6. On the *Data Source* tab, enter the user name and password for the database.
7. On the *TCP/IP* tab, enter the database name, a name for the alias, the host name, and the port number, and then click **OK**.



Database name: database1

Database alias: alias1

Host name: host1

Port number: 50000

□ The database physically resides on a host or OS/400 system.

● Connect directly to the server

○ Connect to the server via the gateway

□ DCS Parameters: ,,INTERRUPT_ENABLED,,,,,

Optimize for application:

8. Enter the username and password again, and then click **OK**.

## 11.2.10.5 IBM DB2 for i (JDBC)

This example illustrates how to connect to an IBM DB2 for i database server via JDBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC driver (one or several .jar files that provide connectivity to the database) must be available on your operating system. This example uses the open source **Toolbox for Java/JTOpen** version 9.8 (http://jt400.sourceforge.net/). After you download the package and unpack to a local directory, the required .jar files will be available in the **lib** subdirectory.
- You have the following database connection details: host, port, database path, name, or alias, username, and password.

### Connection

1. Start the database connection wizard 961 and click **JDBC Connections** (*see JDBC Connection 979 for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH 981 of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.
4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the IBM DB2 for i example*

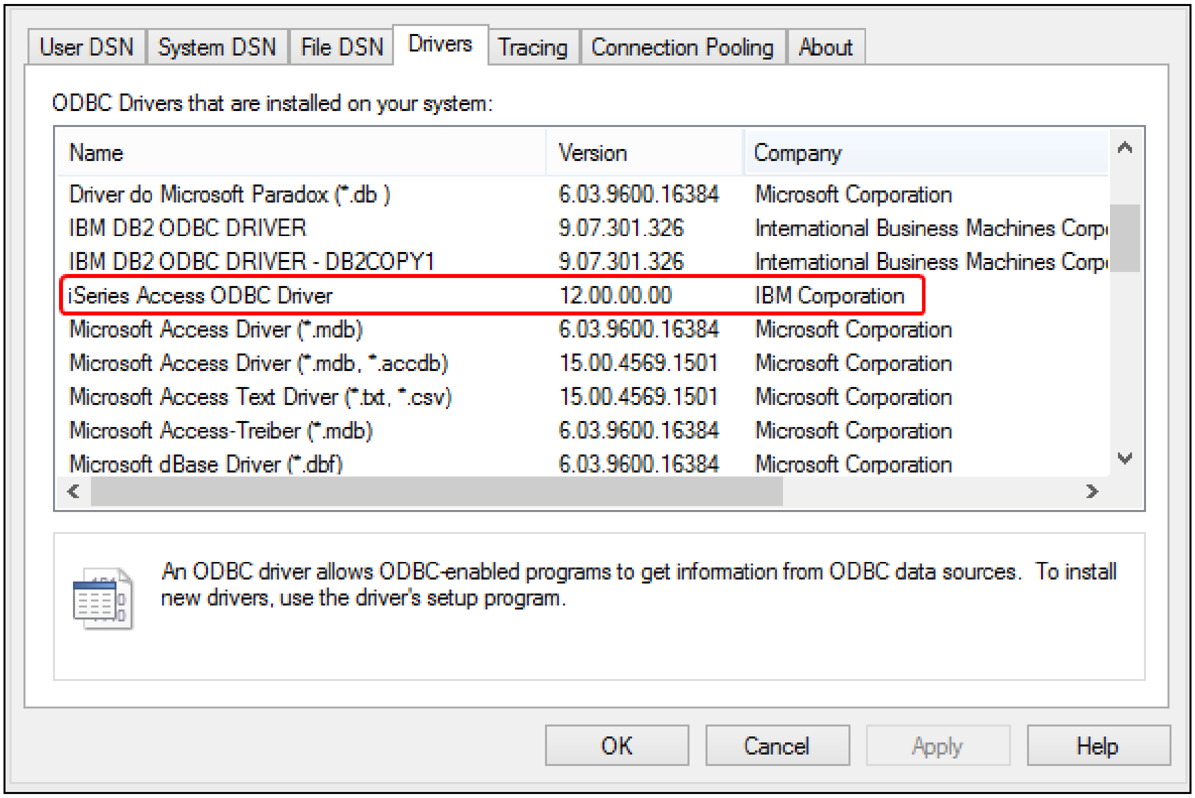
| Field | Value |
|---|---|
| Classpaths | `C:\jdbc\jtopen_9_8\jt400.jar` |
| Driver | *com.ibm.as400.access.AS400JDBCDriver* |
| Database URL | `jdbc:as400://host[:<port>]/<database path or alias>` |

## 11.2.10.6 IBM DB2 for i (ODBC)

This example illustrates how to connect to an *IBM DB2 for i* database via ODBC.

### Prerequisites

- *IBM System i Access for Windows* must be installed on your operating system. For installation instructions, check the documentation supplied with your *IBM DB2 for i* software. After installation, check if the ODBC driver is available on your machine[984].



ODBC Drivers that are installed on your system:

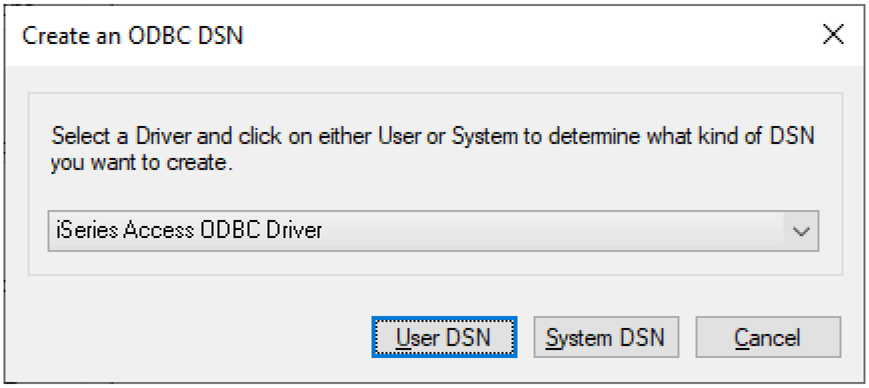
| Name | Version | Company |
|---|---|---|
| Driver do Microsoft Paradox (*.db ) | 6.03.9600.16384 | Microsoft Corporation |
| IBM DB2 ODBC DRIVER | 9.07.301.326 | International Business Machines Corp |
| IBM DB2 ODBC DRIVER - DB2COPY1 | 9.07.301.326 | International Business Machines Corp |
| iSeries Access ODBC Driver | 12.00.00.00 | IBM Corporation |
| Microsoft Access Driver (*.mdb) | 6.03.9600.16384 | Microsoft Corporation |
| Microsoft Access Driver (*.mdb, *.accdb) | 15.00.4569.1501 | Microsoft Corporation |
| Microsoft Access Text Driver (*.txt, *.csv) | 15.00.4569.1501 | Microsoft Corporation |
| Microsoft Access-Treiber (*.mdb) | 6.03.9600.16384 | Microsoft Corporation |
| Microsoft dBase Driver (*.dbf) | 6.03.9600.16384 | Microsoft Corporation |

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

- You have the following database connection details: the IP address of the database server, and the database user name and password.
- Run System i Navigator and follow the wizard to create a new connection. When prompted to specify a system, enter the IP address of the database server. After creating the connection, verify the connection by click it and selecting **File | Diagnostics | Verify Connection**.

### Connection

1. Start the database connection wizard[961] and click **ODBC Connections**.
2. Click *User DSN*. and click **Create a New DSN** + .
3. Select *iSeries Access ODBC Driver* from the list, and click **User DSN** (or **System DSN** if applicable).

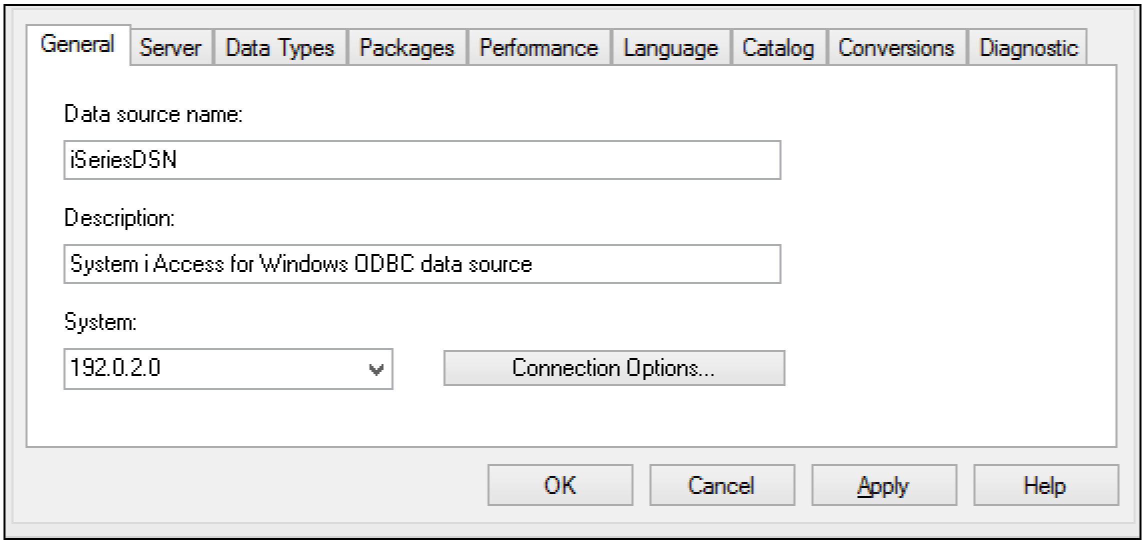


Create an ODBC DSN

Select a Driver and click on either User or System to determine what kind of DSN you want to create.

iSeries Access ODBC Driver

4. Enter a data source name and select the connection from the System combo box. In this example, the data source name is iSeriesDSN and the System is 192.0.2.0.

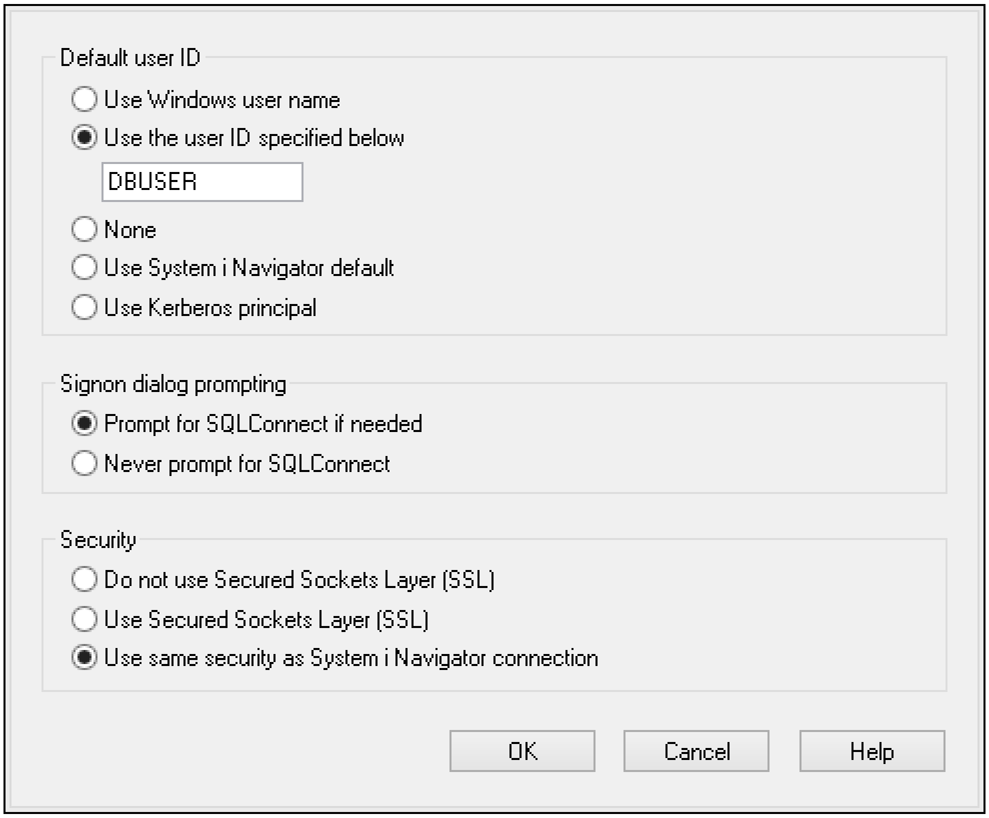


Data source name: iSeriesDSN

Description: System i Access for Windows ODBC data source

System: 192.0.2.0

5. Click **Connection Options**, select *Use the User ID Specified Below* and enter the name of the database user (in this example, *DBUSER*).



Default user ID

○ Use Windows user name

● Use the user ID specified below

DBUSER

○ None

○ Use System i Navigator default

○ Use Kerberos principal

Signon dialog prompting

● Prompt for SQLConnect if needed

○ Never prompt for SQLConnect

Security

○ Do not use Secured Sockets Layer (SSL)

○ Use Secured Sockets Layer (SSL)

● Use same security as System i Navigator connection

6. Click **OK**. The new data source becomes available in the list of DSNs.
7. Click **Connect**.
8. Enter the user name and password to the database when prompted, and then click **OK**.

## 11.2.10.7 IBM Informix (JDBC)

This example illustrates how to connect to an IBM Informix database server via JDBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC driver (one or several .jar files that provide connectivity to the database) must be available on your operating system. In this example, IBM Informix JDBC driver version 3.70 is used. For the driver's installation instructions, see the documentation accompanying the driver or the "IBM Informix JDBC Driver Programmer's Guide").
- You have the following database connection details: host, name of the Informix server, database, port, username, and password.

## Connection

1. Start the database connection wizard 961 and click **JDBC Connections** (*see JDBC Connection 979 for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH 981 of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.
4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the IBM Informix example*

| **Field** | **Value** |
|---|---|
| Classpaths | `C:\Informix_JDBC_Driver\lib\ifxjdbc.jar` |
| Driver | *com.informix.jdbc.IfxDriver* |
| Database URL | `jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver;` |

## 11.2.10.8 MariaDB (ODBC)

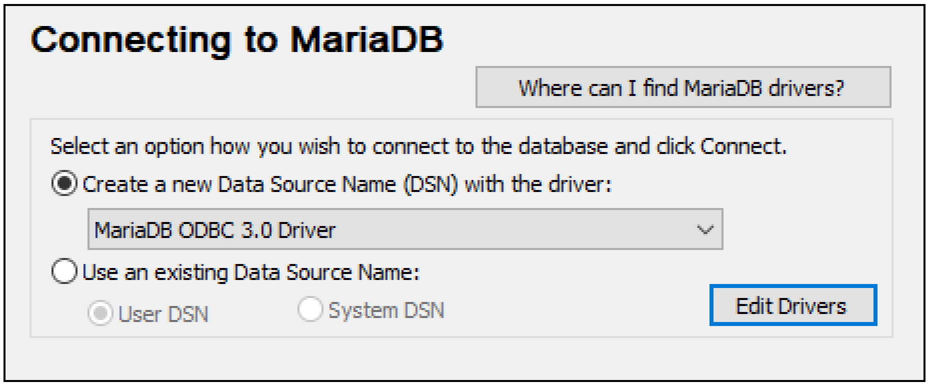
This example illustrates how to connect to a MariaDB database server via ODBC.

## Prerequisites

- The MariaDB ODBC Connector must be installed.
- You have the following database connection details: host, database, port, username, and password.

## Connection

1. Start the database connection wizard 961, select **Connection Wizard** and then *MariaDB (ODBC/Native)*. Click **Next**.
2. Select *Create a new Data Source Name (DSN) with the driver*, and choose *MariaDB ODBC 3.0 Driver*. (If no such driver is available in the list, click **Edit Drivers**, and select any available MariaDB driver from the the list of ODBC drivers installed on your system.) Click **Connect**. to start the New Data Source Wizard.



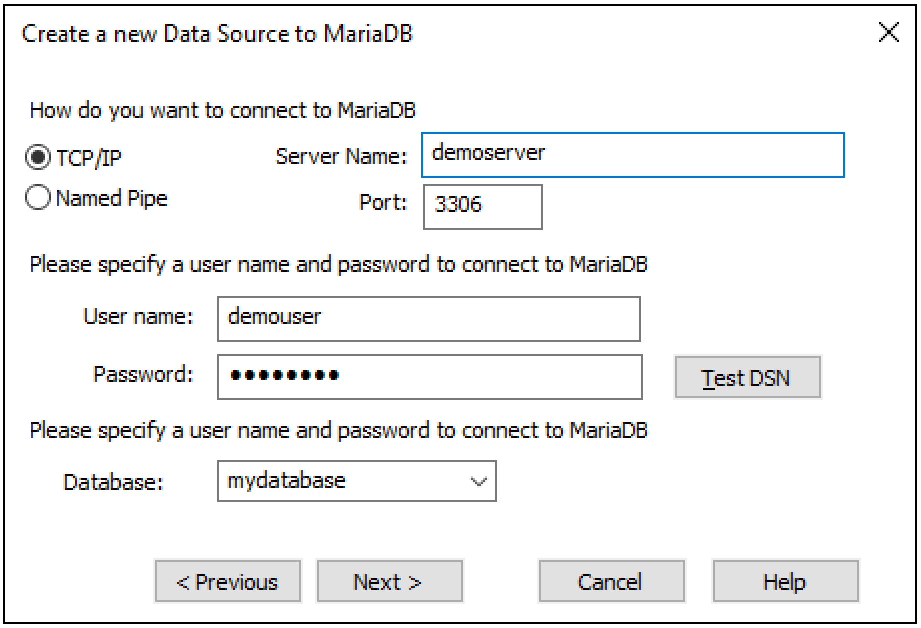
Connecting to MariaDB

Select an option how you wish to connect to the database and click Connect.

● Create a new Data Source Name (DSN) with the driver: MariaDB ODBC 3.0 Driver

○ Use an existing Data Source Name: ● User DSN ○ System DSN

3. In the first screen of the wizard, enter a name for the data source, optionally, a description to help you identify this ODBC data source in the future, and click **OK**.
4. In the next screen (screenshot below), fill in the database connection credentials (TCP/IP Server, User Name, and Password), select a database, and click **Test DSN**.



Create a new Data Source to MariaDB

How do you want to connect to MariaDB

● TCP/IP  Server Name: demoserver

○ Named Pipe  Port: 3306

Please specify a user name and password to connect to MariaDB

User name: demouser

Password: ••••••••

Please specify a user name and password to connect to MariaDB

Database: mydatabase

5. Upon successful connection, a message to that effect is displayed. Click **Next** and complete the wizard. Other parameters may be required, for example, SSL certificates if you are connecting to MariaDB via a secure connection.

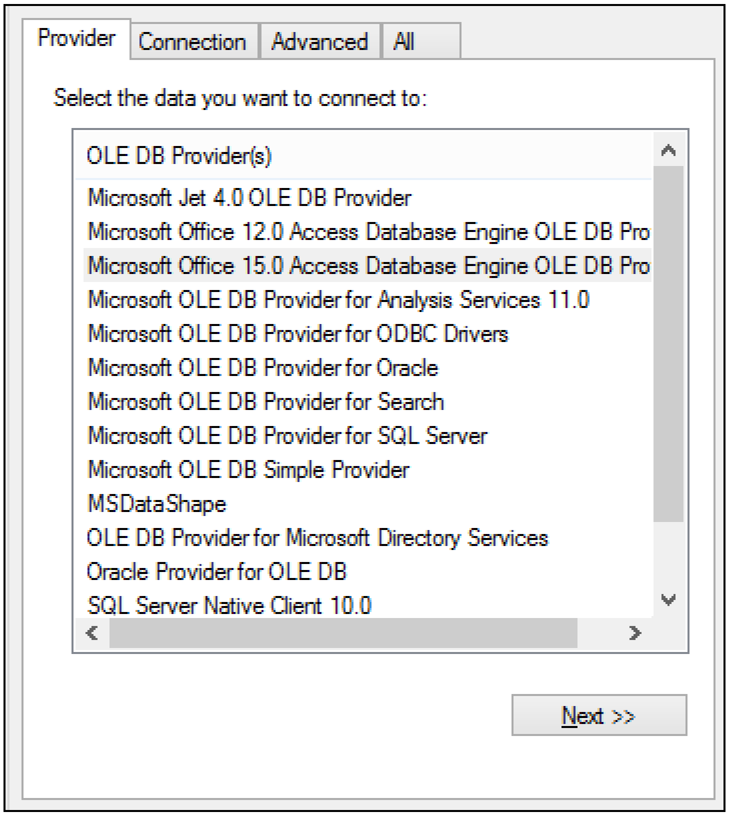
**Note:** If the database server is remote, it must be configured by the server administrator to accept remote connections from your machine's IP address.

## 11.2.10.9 Microsoft Access (ADO)

A simple way to connect to a Microsoft Access database is to follow the wizard and browse for the database file, as shown in Connecting to an Existing Microsoft Access Database[970]. An alternative approach is to set up an ADO connection explicitly, as shown in this topic. This approach is useful if your database is password-protected. (It is also possible to connect to Microsoft Access through an ODBC connection, but it has limitations, so it is best to avoid it.)

## Connection

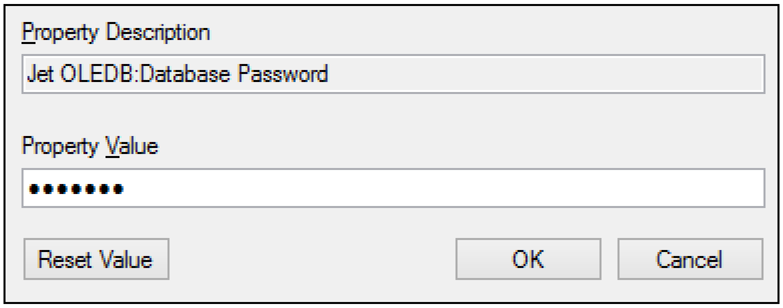
1. Start the database connection wizard 961.
2. Click **ADO Connections**, then click **Build**.
3. In the dialog that appears (*screenshot below*), go to the *Provider* tab, select *Microsoft Office 15.0 Access Database Engine OLE DB Provider*, and then click **Next**.



Select the data you want to connect to:

OLE DB Provider(s)
Microsoft Jet 4.0 OLE DB Provider
Microsoft Office 12.0 Access Database Engine OLE DB Pro
Microsoft Office 15.0 Access Database Engine OLE DB Pro
Microsoft OLE DB Provider for Analysis Services 11.0
Microsoft OLE DB Provider for ODBC Drivers
Microsoft OLE DB Provider for Oracle
Microsoft OLE DB Provider for Search
Microsoft OLE DB Provider for SQL Server
Microsoft OLE DB Simple Provider
MSDataShape
OLE DB Provider for Microsoft Directory Services
Oracle Provider for OLE DB
SQL Server Native Client 10.0

4. Go to the dialog's *Connection* tab, and in the *Data Source* field, enter the path to the Microsoft Access file in UNC format, for example, `\\myserver\\mynetworkshare\Reports\Revenue.accdb`, where `myserver` is the name of the server and `mynetworkshare` is the name of the network share.
5. On the *All* tab, double click the *Jet OLEDB:Database Password* property and enter the database password as property value (*see screenshot below*)



Property Description
Jet OLEDB:Database Password

Property Value
•••••••

If you are still unable to connect, locate the workgroup information file (`System.MDW`) applicable to your user profile, and set the value of the *Jet OLEDB: System database* property to the path of the `System.MDW` file.

## 11.2.10.10 Microsoft Azure SQL (ODBC)

In order to connect properly to an Azure SQL database, you must install the latest SQL Server Native Client.

For information about connecting to an Azure SQL database in the cloud, see this Altova blog entry.

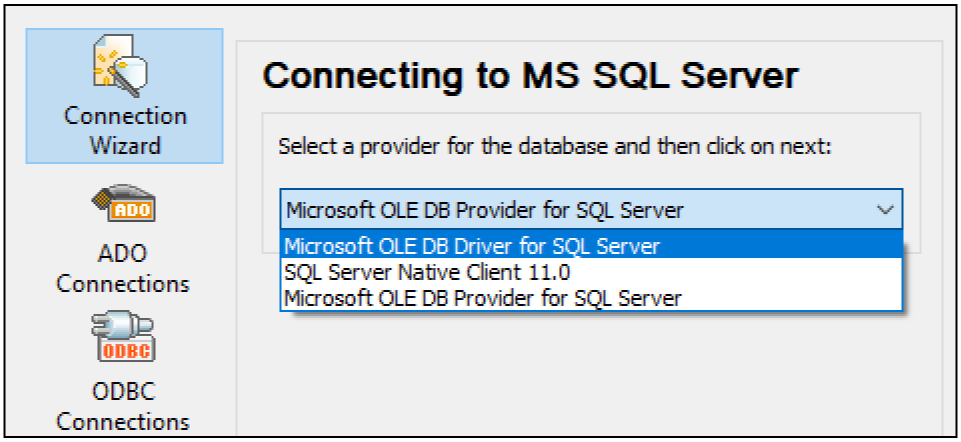
## 11.2.10.11 Microsoft SQL Server (ADO)

This example illustrates how to connect to a SQL Server database through ADO. These instructions work with the recommended Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) that matches your MobileTogether Designer platform (32-bit or 64-bit). For other ADO providers, the instructions would be similar but you may need to set additional connection properties as described in Setting up the SQL Server Data Link Properties 971 .

**Note:** The *Microsoft OLE DB Provider for SQL Server (SQLOLEDB)* is known to have issues with parameter binding of complex queries like Common Table Expressions (CTE) and nested SELECT statements.

### Connection

1. Start the database connection wizard 961 , select **Connection Wizard** and then *Microsoft SQL Server (ADO)*. Click **Next**.
2. A list of available ADO providers is displayed (*screenshot below*). In this example, we use *Microsoft OLE DB Driver for SQL Server*. If it's not in the list, make sure that it is installed on your computer.



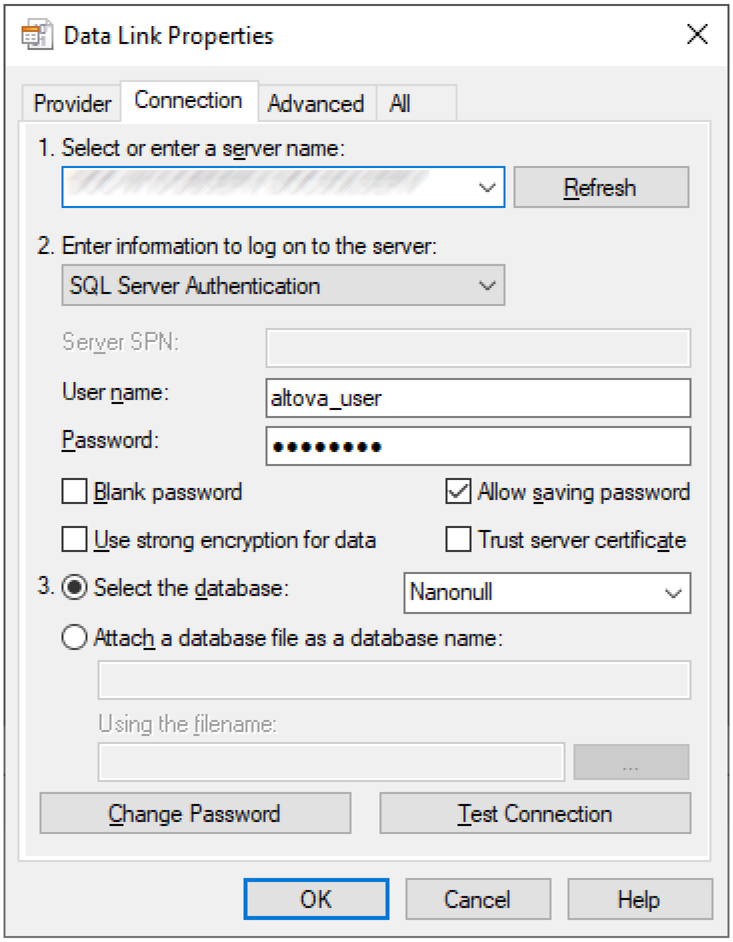
Connecting to MS SQL Server

Select a provider for the database and then click on next:

Microsoft OLE DB Provider for SQL Server

Microsoft OLE DB Driver for SQL Server
SQL Server Native Client 11.0
Microsoft OLE DB Provider for SQL Server

3. Click **Next**. The Data Link Properties dialog appears (*screenshot below*).



1. Select or enter a server name: [redacted]

2. Enter information to log on to the server: SQL Server Authentication

Server SPN:

User name: altova_user

Password: •••••••••

□ Blank password ☑ Allow saving password

□ Use strong encryption for data □ Trust server certificate

3. ● Select the database: Nanonull

○ Attach a database file as a database name:

Using the filename:

4. In the *Connection* tab, select or enter the name of the database server, for example, *SQLSERV01*. If you are connecting to a named SQL Server instance, the server name will be something like *SQLSERV01\SOMEINSTANCE*.
5. If the database server was configured to allow connections from users authenticated on the Windows domain, select Windows Authentication. Otherwise, select SQL Server Authentication, clear the Blank password check box, and enter the database credentials in the relevant boxes.
6. Select the *Allow saving password* check box and the database to which you are connecting (in this example, *Nanonull*).
7. You can test the connection by clicking **Test Connection**.
8. Click **OK** when done.

## 11.2.10.12 Microsoft SQL Server (ODBC)

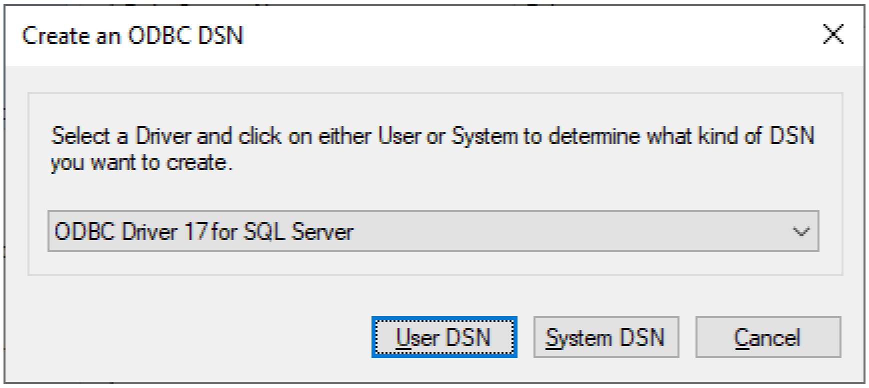
This example illustrates how to connect to a SQL Server database via ODBC.

### Prerequisites

Download and install the Microsoft ODBC Driver for SQL Server from the Microsoft website. This example uses Microsoft ODBC Driver 17 for SQL Server to connect to a SQL Server 2016 database. You might need a different ODBC driver version, depending on your SQL Server version. For information about compatibility, see the driver's system requirements.

## Connection

1. Start the database connection wizard 961 and click **ODBC Connections**.
2. Select *User DSN* (or *System DSN* if you have administrative privileges), and then click **Create a New DSN** ✚ .
3. Select the driver from the list. Note that the driver appears in the list only after it has been installed.

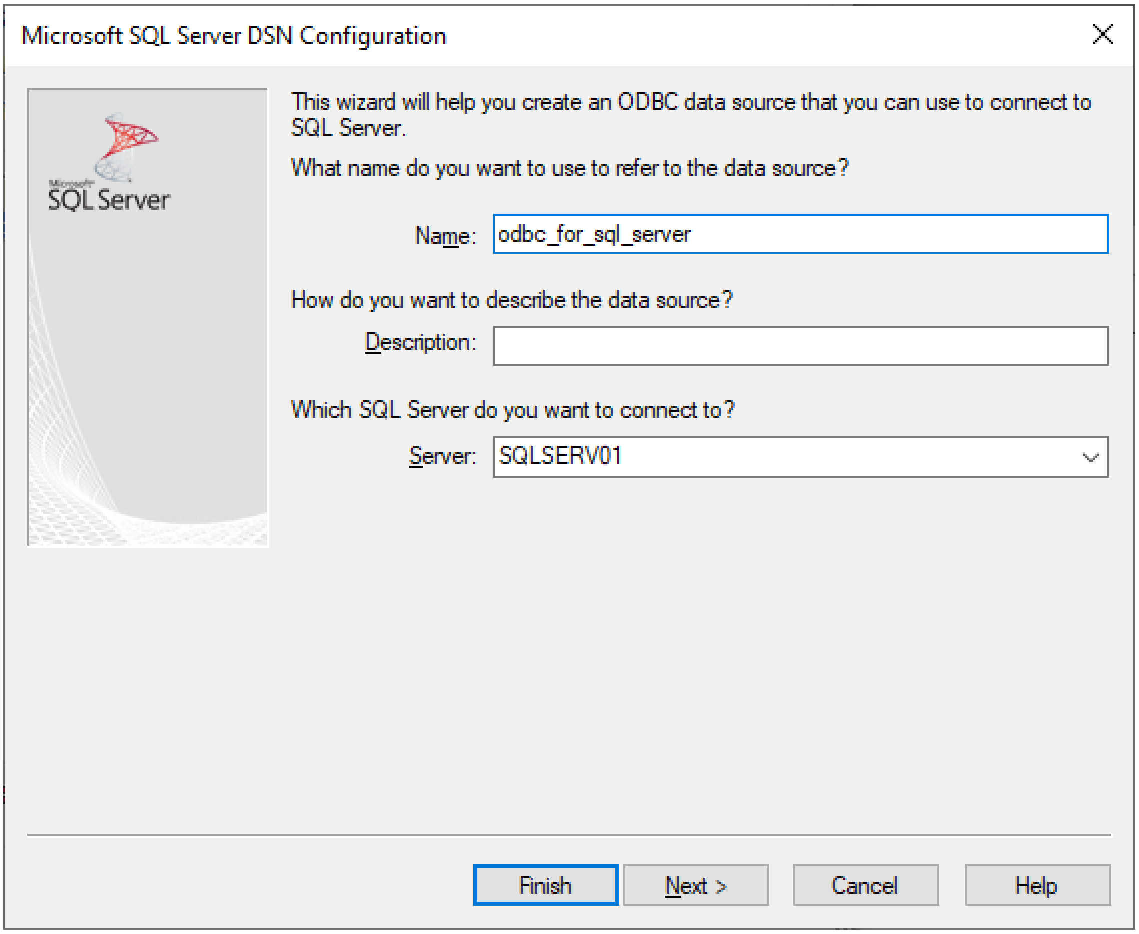


Create an ODBC DSN

Select a Driver and click on either User or System to determine what kind of DSN you want to create.

ODBC Driver 17 for SQL Server

4. Click *User DSN* (or *System DSN* if you are creating a System DSN). Creating a System DSN requires that MobileTogether Designer be run as an administrator. Therefore, in order to create a System DSN, cancel the wizard, make sure that you run MobileTogether Designer as an administrator, and perform the steps above again.
5. Enter a name for the data source, enter, optionally, a description to identify this connection, and select from the list the SQL Server to which you are connecting. Click **Next** to go to the next screen.



This wizard will help you create an ODBC data source that you can use to connect to SQL Server.

What name do you want to use to refer to the data source?

Name: odbc_for_sql_server

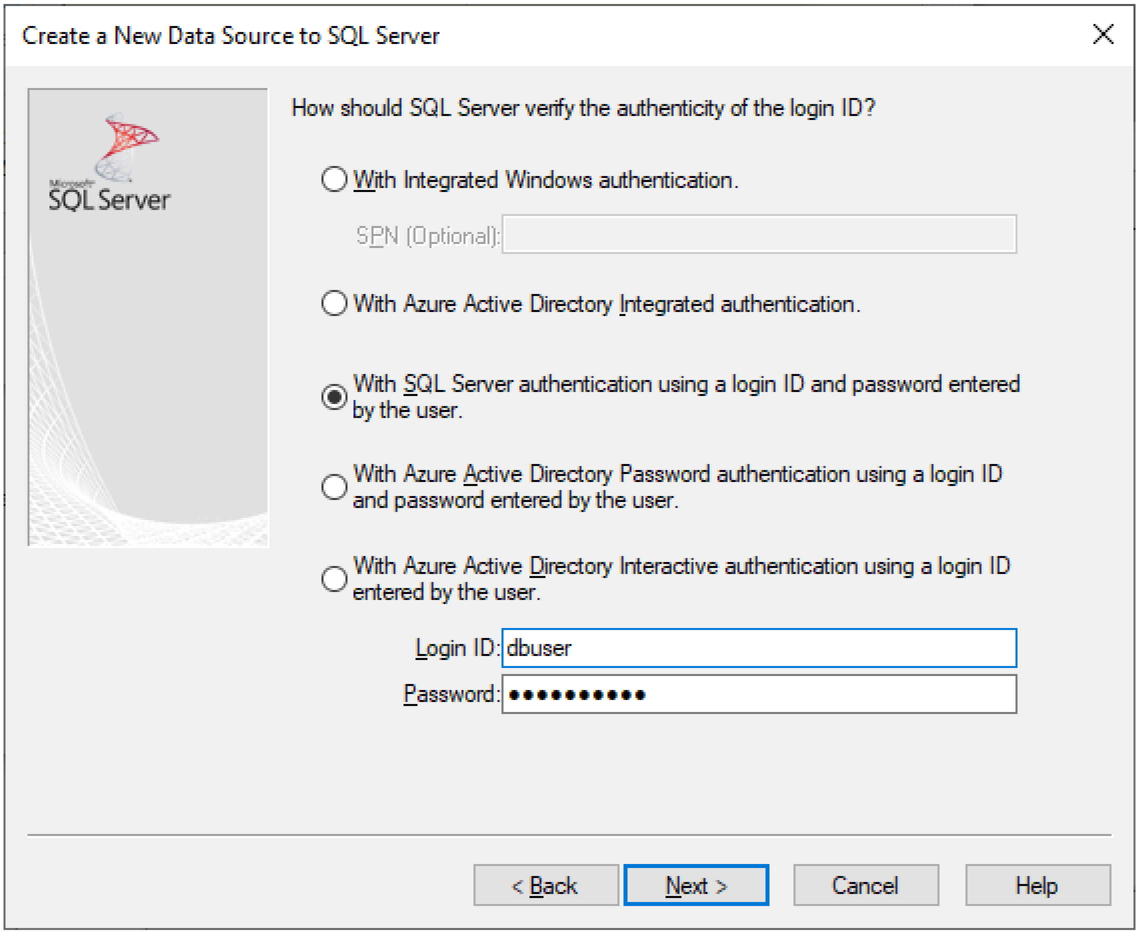
How do you want to describe the data source?

Description:

Which SQL Server do you want to connect to?

Server: SQLSERV01

6. In the next screen, select the authentication method to use. If the database server was configured to allow connections from users authenticated on the Windows domain, select *With Integrated Windows authentication*. Otherwise, select one of the other options, as applicable. This example uses *With SQL Server Authentication*, which requires that the user name and password be entered in the relevant boxes. Click **Next** when done.



Create a New Data Source to SQL Server

How should SQL Server verify the authenticity of the login ID?

○ With Integrated Windows authentication.

SPN (Optional):

○ With Azure Active Directory Integrated authentication.

● With SQL Server authentication using a login ID and password entered by the user.

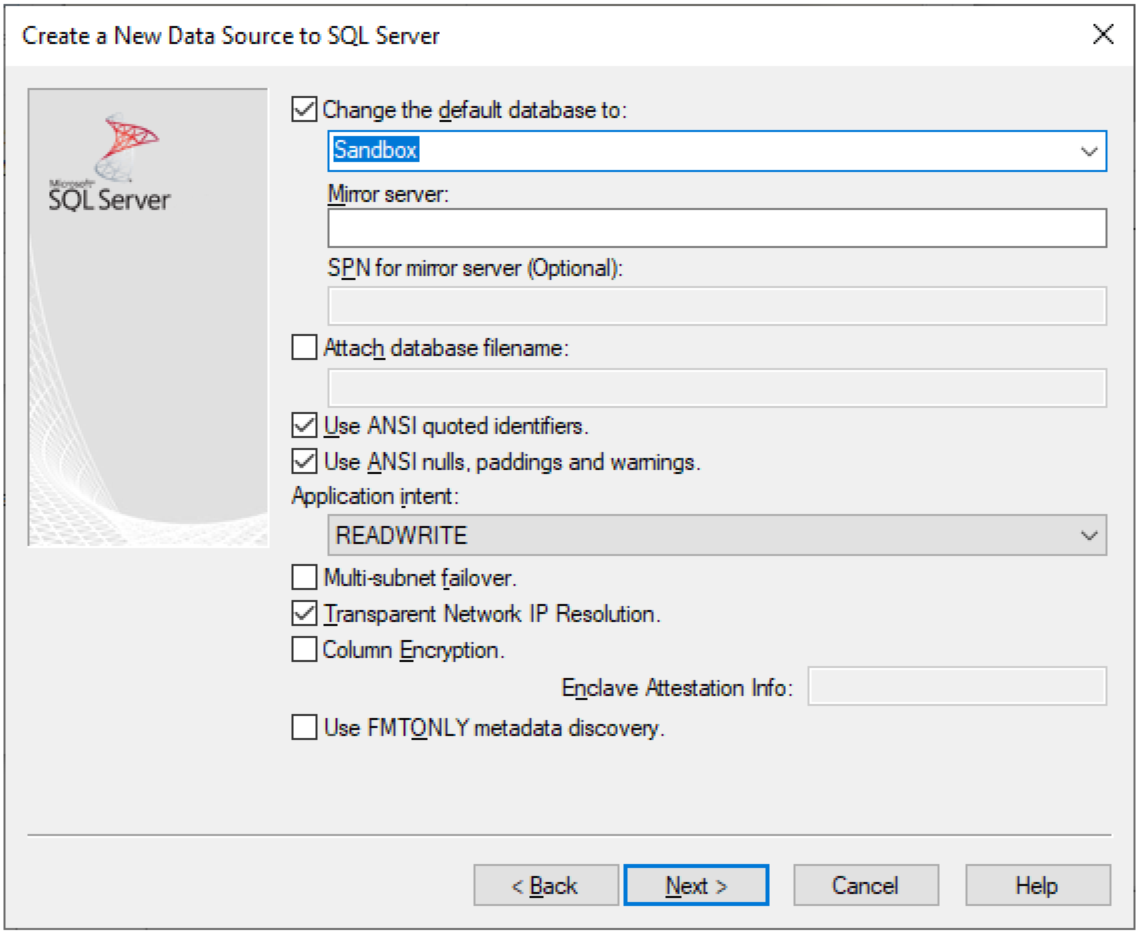
○ With Azure Active Directory Password authentication using a login ID and password entered by the user.

○ With Azure Active Directory Interactive authentication using a login ID entered by the user.

Login ID: dbuser

Password: ●●●●●●●●●●

7. In the next screen of our example (*screenshot below*), we have selected *Change the default database* and entered *Sandbox* (the name of the database to which to connect). Click **Next** after you have entered the settings you want.



Create a New Data Source to SQL Server

☑ Change the default database to: Sandbox

Mirror server:

SPN for mirror server (Optional):

□ Attach database filename:

☑ Use ANSI quoted identifiers.

☑ Use ANSI nulls, paddings and warnings.

Application intent: READWRITE

□ Multi-subnet failover.

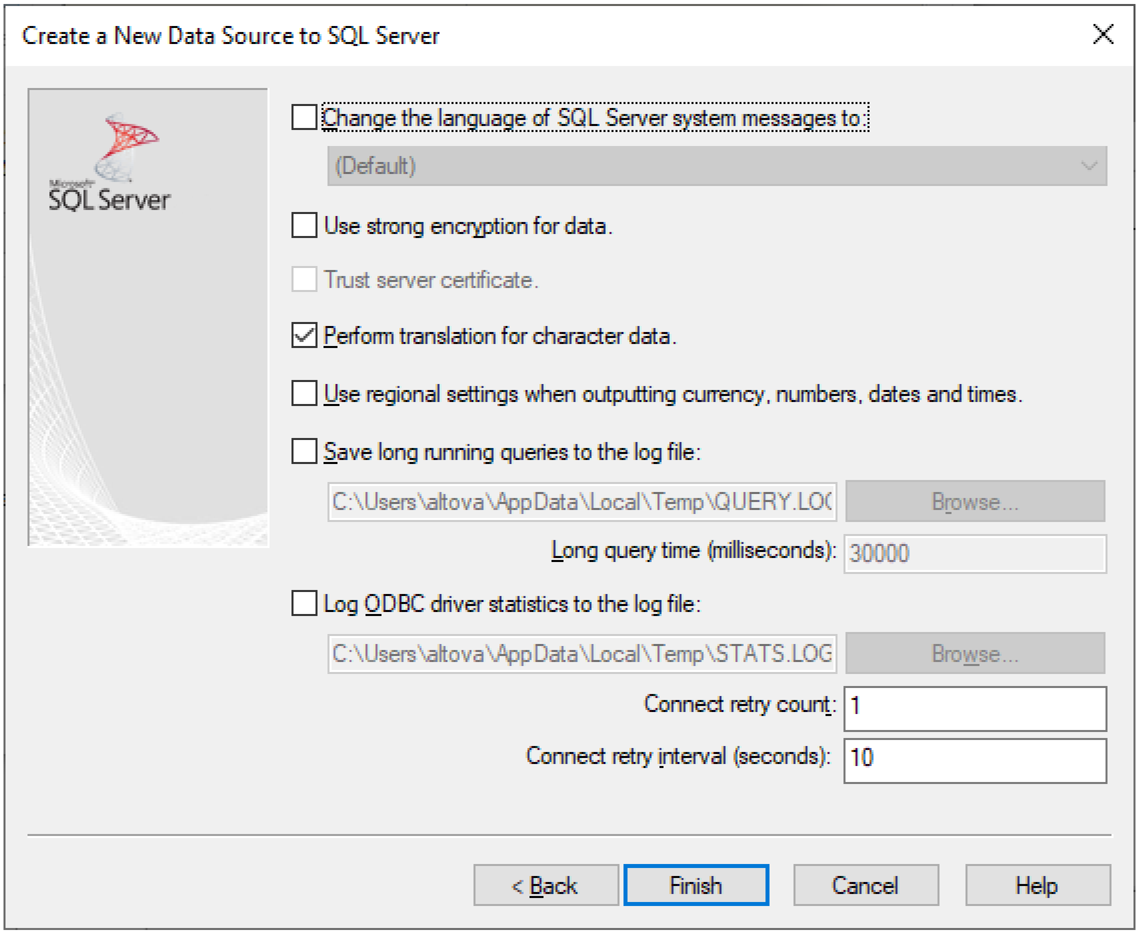
☑ Transparent Network IP Resolution.

□ Column Encryption.

Enclave Attestation Info:

□ Use FMTONLY metadata discovery.

8. Click **Next** and, optionally, configure additional parameters for this connection. Click **Finish** when done.



Create a New Data Source to SQL Server

☐ Change the language of SQL Server system messages to:

(Default)

☐ Use strong encryption for data.

☐ Trust server certificate.

☑ Perform translation for character data.

☐ Use regional settings when outputting currency, numbers, dates and times.

☐ Save long running queries to the log file:

C:\Users\altova\AppData\Local\Temp\QUERY.LO(

Long query time (milliseconds): 30000

☐ Log ODBC driver statistics to the log file:

C:\Users\altova\AppData\Local\Temp\STATS.LOG

Connect retry count: 1

Connect retry interval (seconds): 10

9. A confirmation dialog box listing the connection details opens. Click **OK**.
10. The data source now appears in the list of *User* or *System* data sources.

## 11.2.10.13 MySQL (ODBC)

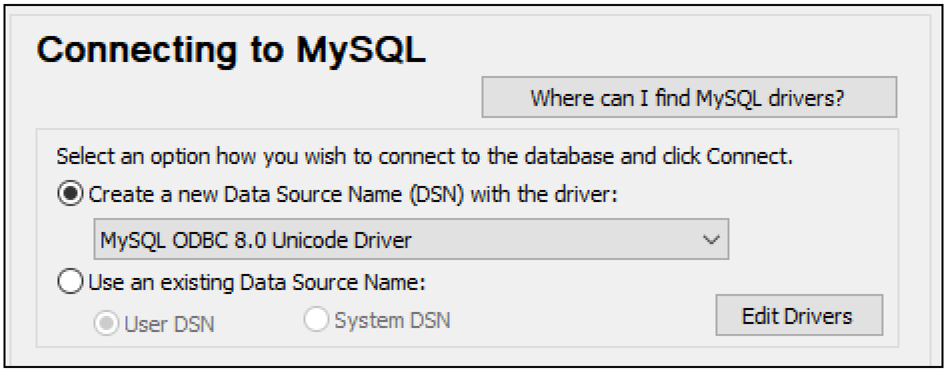
This example illustrates how to connect to a MySQL database server from a Windows machine via the MySQL ODBC, which driver must be downloaded and installed separately. This example uses MySQL Connector/ODBC 8.0.

### Prerequisites

- MySQL ODBC driver must be installed on your operating system. Check the MySQL documentation for the driver version recommended for your database server version.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the MySQL ODBC driver.
- You have the following database connection details: host, database, port, username, and password.

## Connection

1. Start the database connection wizard[961], select **Connection Wizard** and then *MySQL (ODBC/Native)*. Click **Next**.
2. Select *Create a new Data Source Name (DSN) with the driver*, and select a MySQL driver. (If no such driver is available in the list, click **Edit Drivers**, and select any available MySQL driver from the the list of ODBC drivers installed on your system. See also Viewing the Available ODBC Drivers[984].)



**Connecting to MySQL**

Select an option how you wish to connect to the database and click Connect.

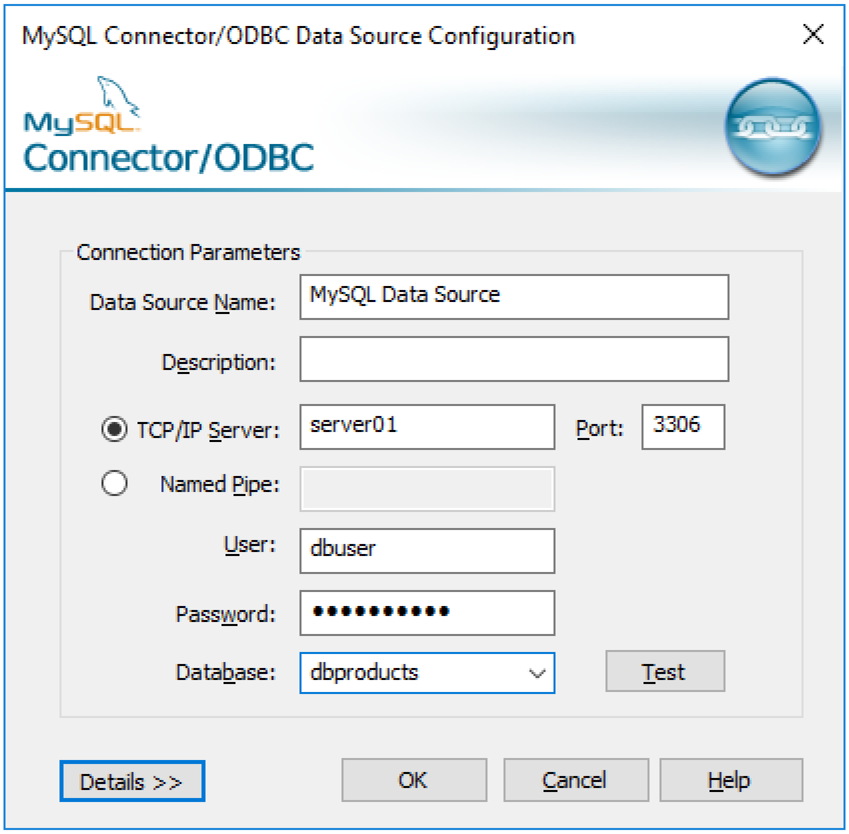
● Create a new Data Source Name (DSN) with the driver:

MySQL ODBC 8.0 Unicode Driver

○ Use an existing Data Source Name:

● User DSN ○ System DSN

3. Click **Connect**. to open the Data Source Configuration dialog (*screenshot below*).



MySQL Connector/ODBC Data Source Configuration

Connection Parameters

Data Source Name: MySQL Data Source

Description:

● TCP/IP Server: server01 Port: 3306

○ Named Pipe:

User: dbuser

Password: ●●●●●●●●●●●

Database: dbproducts

4. In the Data Source Name box, enter a descriptive name that will help you identify this ODBC data source in future. Optionally enter a description.

5. Fill in the database connection credentials (TCP/IP Server, User, Password), select a database, and then click **OK**.

**Note:** If the database server is remote, it must be configured by the server administrator to accept remote connections from your machine's IP address. Also, if you click **Details >>**, there are several additional parameters available for configuration. Check the driver's documentation before changing the default values.

## 11.2.10.14 Oracle (JDBC)

This example shows how to connect to an Oracle database server using the JDBC interface. The connection is created as a pure Java connection, using the Oracle Instant Client Package (Basic) available from the Oracle website. The advantage of this connection type is that it requires only the Java environment and the .jar libraries supplied by the Oracle Instant Client Package.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The Oracle Instant Client Package (Basic) must be available on your operating system. The package can be downloaded from the official Oracle website. This example uses Oracle Instant Client Package version 12.1.0.2.0, for Windows 32-bit and, consequently, Oracle JDK 32-bit.
- You have the following database connection details: host, port, service name, username, and password.

### Connection

1. Start the database connection wizard 961 and click **JDBC Connections** (*see JDBC Connection 979 for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH 981 of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.
4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the Oracle example*

| **Field** | **Value** |
|---|---|
| Classpaths | `C:\jdbc\instantclient_12_1\ojdbc7.jar` |
| Driver | *oracle.jdbc.OracleDriver* or *oracle.jdbcdriver.OracleDriver* |

| | |
|---|---|
| Database URL | `jdbc:oracle:thin:@//host:port:service` |

## 11.2.10.15 Oracle (ODBC)

This example shows how to connect from MobileTogether Designer to an Oracle database server on a network machine, via an Oracle database client installed on the local operating system.

The example includes instructions for setting up an ODBC data source (DSN) using the database connection wizard in MobileTogether Designer. If you have already created a DSN, or if you prefer to create it directly from the *ODBC Data Source Administrator* in Windows, you can do so, and then select it when prompted by the wizard. For more information about ODBC data sources, see Setting up an ODBC Connection[982].

### Prerequisites

- The Oracle database client (which includes the ODBC Oracle driver) must be installed and configured on your system. For instructions, see the Oracle documentation.
- The **`tnsnames.ora`** file located in Oracle home directory contains an entry that describes the database connection parameters, in a format similar to this:

```
ORCL =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = server01)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SID = orcl)
      (SERVER = DEDICATED)
    )
  )
```

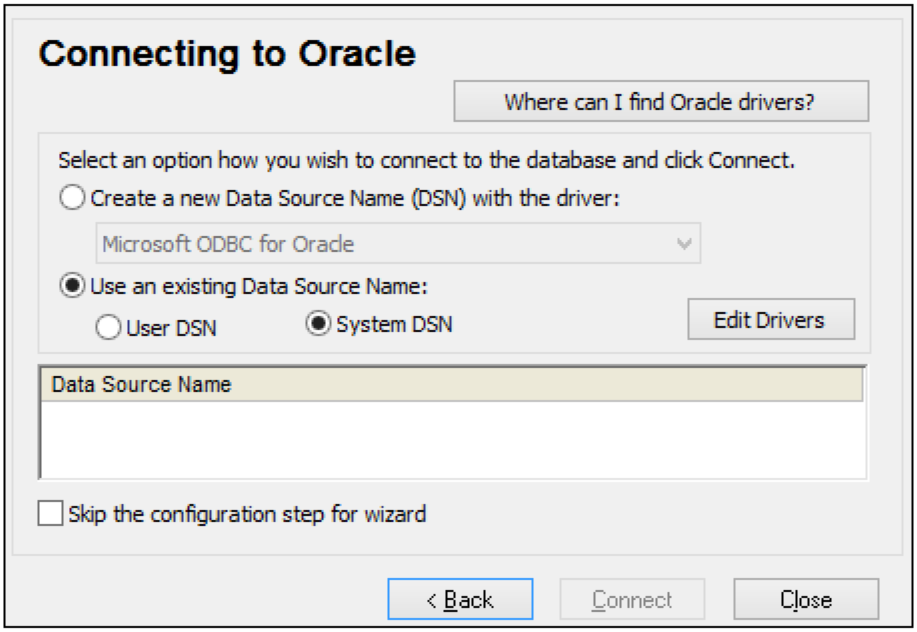
For Oracle database client 11.2.0, the default Oracle home directory path could be as follows:

```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

You can add new entries to **`tnsnames.ora`**, either by pasting the connection details and saving the file, or by running the Oracle Net Configuration Assistant, if available. If you want these values to appear in dropdown lists during the configuration process, then you may need to add the path to the admin folder as a **`TNS_ADMIN`** environment variable.

### Connection

1. Start the database connection wizard[961], select **Connection Wizard** and then *Oracle (ODBC/JDBC)*. Click **Next**.
2. Select *ODBC* and click **Next**.
3. In the dialog that appears (*screenshot below*), click **Edit Drivers**.



Connecting to Oracle

Where can I find Oracle drivers?

Select an option how you wish to connect to the database and click Connect.

○ Create a new Data Source Name (DSN) with the driver:

Microsoft ODBC for Oracle

● Use an existing Data Source Name:

○ User DSN ● System DSN

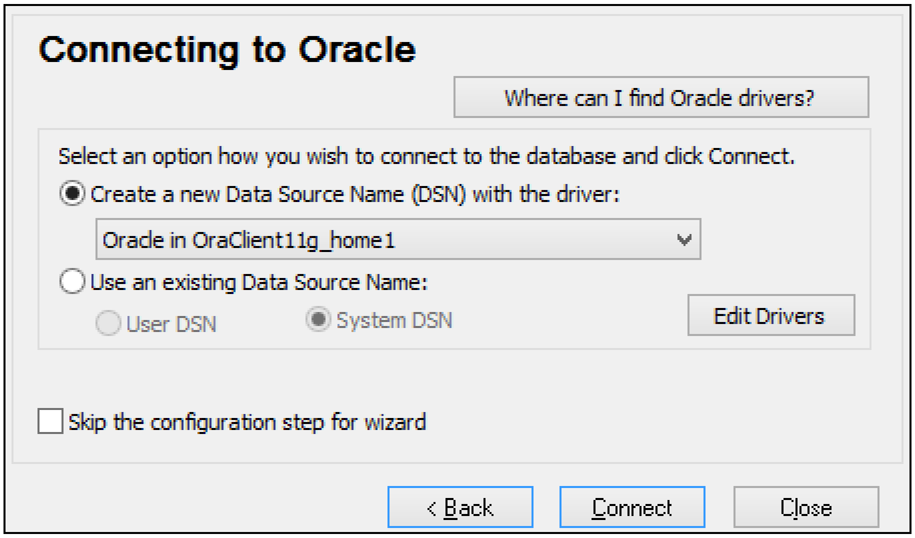
Edit Drivers

Data Source Name

□ Skip the configuration step for wizard

< Back Connect Close

4. From the list of Oracle drivers available on your system, select the Oracle driver you wish to use and click **Back**.
5. Select *Create a new data source name (DSN) with the driver*, and then select the Oracle driver chosen in step 4. (Avoid using the Microsoft-supplied driver called Microsoft ODBC for Oracle driver. Microsoft recommends using the ODBC driver provided by Oracle.)



Connecting to Oracle

Where can I find Oracle drivers?

Select an option how you wish to connect to the database and click Connect.

● Create a new Data Source Name (DSN) with the driver:

Oracle in OraClient11g_home1

○ Use an existing Data Source Name:

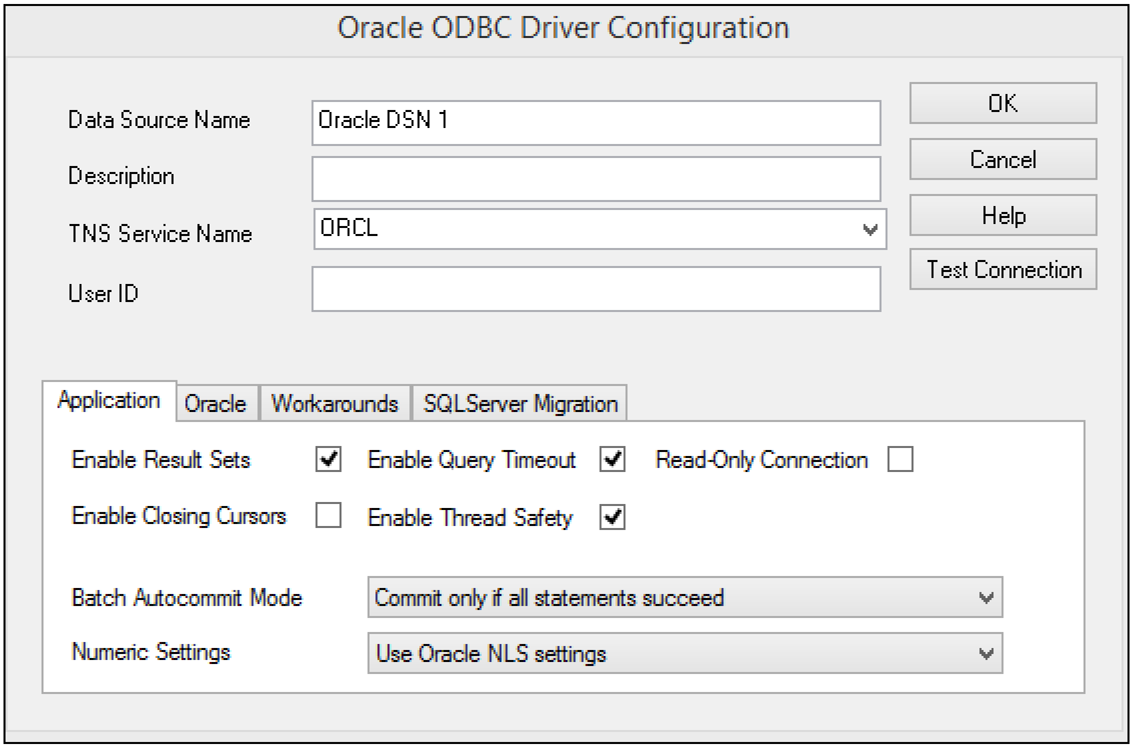
○ User DSN ● System DSN

Edit Drivers

□ Skip the configuration step for wizard

< Back Connect Close

6. Click **Connect**.
7. In the dialog that appears fill in the required fields (*shown filled in the screenshot below*). The connection name must be entered in the *TNS Service Name* field as it is defined in the `tnsnames.ora` file (see *Prerequisites* above). *Note:* If you wish to have the dropdown list of the combo box automatically filled with the values of the **`tnsnames.ora`** file, then you may need to add the path to the admin folder as a **`TNS_ADMIN`** environment variable.



Data Source Name: Oracle DSN 1

Description:

TNS Service Name: ORCL

User ID:

Enable Result Sets ☑ Enable Query Timeout ☑ Read-Only Connection ☐

Enable Closing Cursors ☐ Enable Thread Safety ☑

Batch Autocommit Mode: Commit only if all statements succeed

Numeric Settings: Use Oracle NLS settings

8. Click **OK** when done.
9. In the dialog that appears, enter the username and password to the database, and then click **OK**.

## 11.2.10.16 PostgreSQL (ODBC)

This example shows how to connect to a PostgreSQL database server from a Windows machine via a PostgreSQL ODBC driver installed on the local machine. This example uses the *psqlODBC* driver (version 11.0) downloaded from the official website (see also Database Drivers Overview 963).

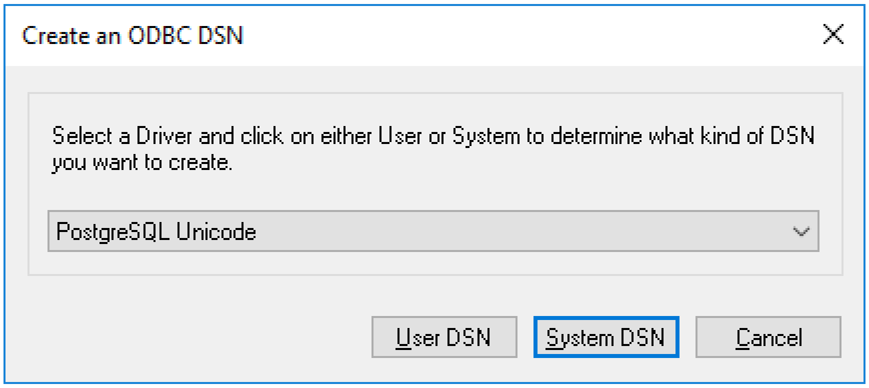
**Note:** You can also connect to a PostgreSQL database server directly (without the ODBC driver), see Native Connections 987.

### Prerequisites

- *psqlODBC* driver must be installed on your system.
- You have the following database connection details: server, port, database, user name, and password.

### Connection

1. Start the database connection wizard 961 and click **ODBC Connections**.
2. Select *User DSN* (or *System DSN* if you have administrative privileges), and then click **Create a New DSN** + .
3. Select the driver from the drop-down list (*screenshot below*) and click **User DSN**. (If no PostgreSQL driver is available in the list, make sure that the PostgreSQL ODBC driver is installed on your operating system.)

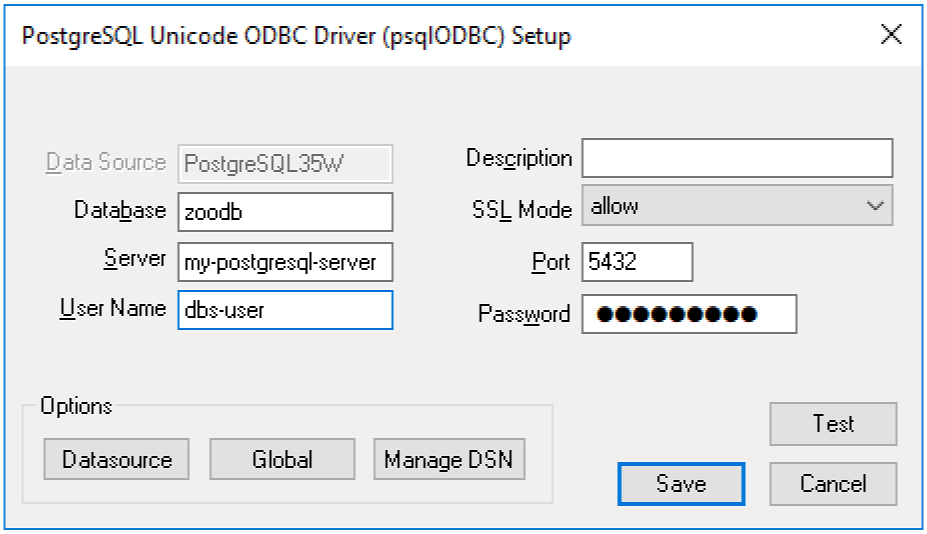


Create an ODBC DSN

Select a Driver and click on either User or System to determine what kind of DSN you want to create.

PostgreSQL Unicode

4. In the dialog that appears (*screenshot below*), fill in the database connection (supplied by the database administrator).



PostgreSQL Unicode ODBC Driver (psqlODBC) Setup

Data Source: PostgreSQL35W

Database: zoodb

Server: my-postgresql-server

User Name: dbs-user

Description:

SSL Mode: allow

Port: 5432

Password: ●●●●●●●●●

5. Click **Save**.

The connection will become now available in the list of ODBC connections. To connect to the database, you can either double-click the connection or select it and then click **Connect**.

## 11.2.10.17 Progress OpenEdge (JDBC)

This example illustrates how to connect to a Progress OpenEdge 11.6 database server via JDBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.

- The operating system's `PATH` environment variable must include the path to the `bin` directory of the JRE or JDK installation directory, for example `C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- The Progress OpenEdge JDBC driver must be available on your operating system. In this example, JDBC connectivity is provided by the openedge.jar and pool.jar driver component files available in `C:\Progress\OpenEdge\java` as part of the OpenEdge SDK installation.
- You have the following database connection details: host, port, database path, name, or alias, username, and password.

### Connection

1. Start the database connection wizard 961 and click **JDBC Connections** (*see JDBC Connection 979 for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH 981 of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.
4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the Progress OpenEdge example*

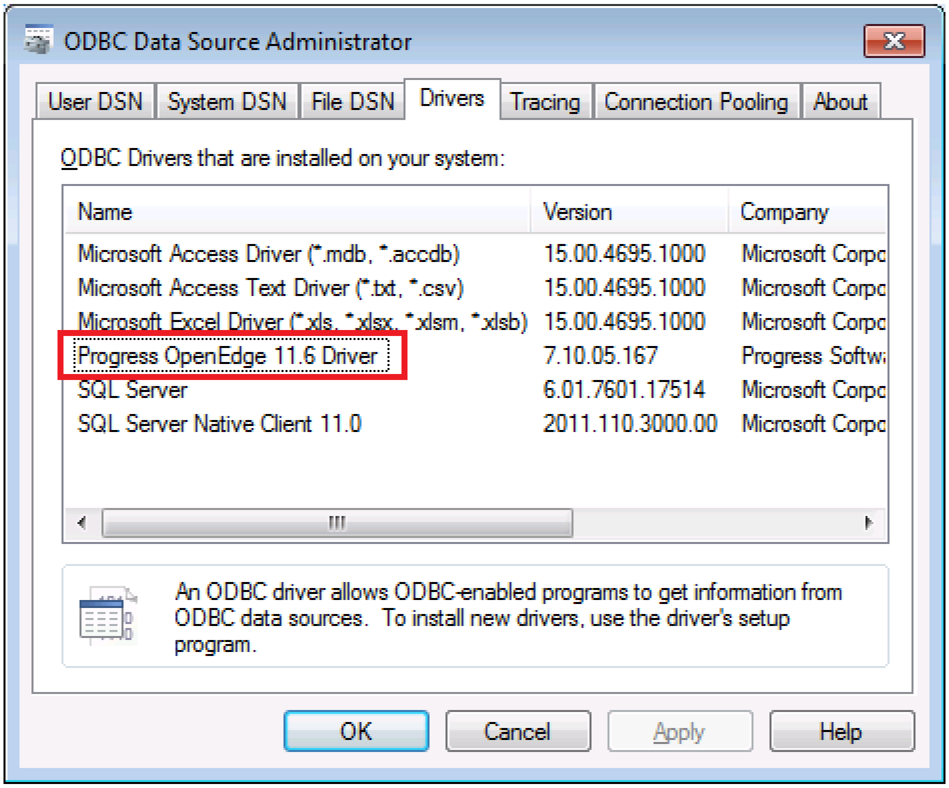
| **Field** | **Value** |
|---|---|
| Classpaths | `C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;` |
| Driver | *com.ddtek.jdbc.openedge.OpenEdgeDriver* |
| Database URL | `jdbc:datadirect:openedge://host:port;databaseName=db_name` |

## 11.2.10.18 Progress OpenEdge (ODBC)

This example shows how to connect to a Progress OpenEdge database server via the Progress OpenEdge 11.6 ODBC driver.

### Prerequisites

- The *ODBC Connector for Progress OpenEdge* driver must be installed on your system (see also Database Drivers Overview 963). Download the version that corresponds to your version of MobileTogether Designer (32-bit or 64-bit). After installation, check if the ODBC driver is available on your machine 984.



ODBC Drivers that are installed on your system:

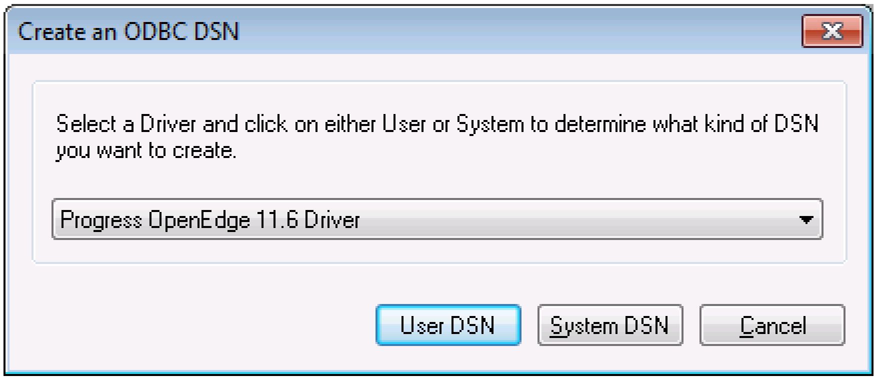
| Name | Version | Company |
|---|---|---|
| Microsoft Access Driver (*.mdb, *.accdb) | 15.00.4695.1000 | Microsoft Corpo |
| Microsoft Access Text Driver (*.txt, *.csv) | 15.00.4695.1000 | Microsoft Corpo |
| Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb) | 15.00.4695.1000 | Microsoft Corpo |
| Progress OpenEdge 11.6 Driver | 7.10.05.167 | Progress Softw |
| SQL Server | 6.01.7601.17514 | Microsoft Corpo |
| SQL Server Native Client 11.0 | 2011.110.3000.00 | Microsoft Corpo |

An ODBC driver allows ODBC-enabled programs to get information from ODBC data sources. To install new drivers, use the driver's setup program.

- You have the following database connection details: host name, port number, database name, user ID, and password.

## Connection

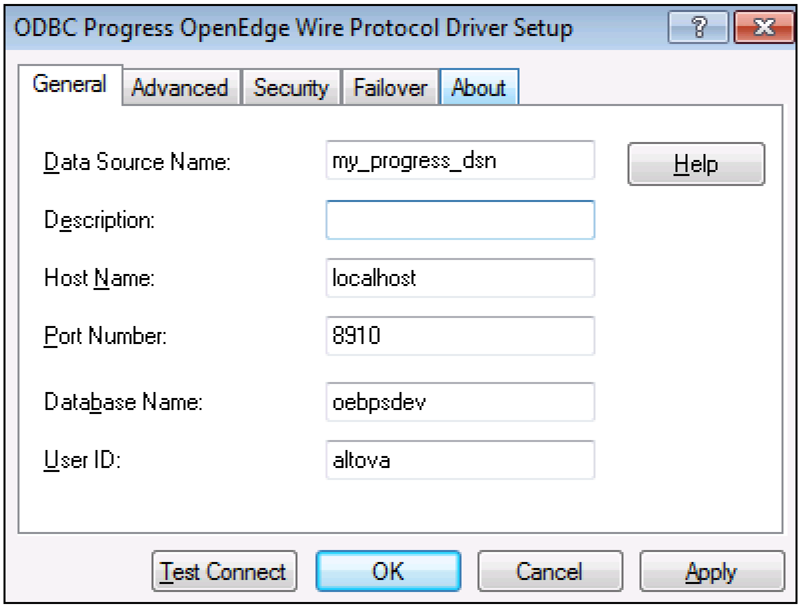
1. Start the database connection wizard[961] and click **ODBC Connections**.
2. Select *User DSN* (or *System DSN* if you have administrative privileges), and then click **Create a New DSN** ✚ .
3. Select *Progress OpenEdge Driver* from the drop-down list (*screenshot below*) and click **User DSN** (or **System DSN** if applicable).



Select a Driver and click on either User or System to determine what kind of DSN you want to create.

Progress OpenEdge 11.6 Driver

4. Fill in the database connection details (Database, Server, Port, User Name, Password), and click **OK**.



Data Source Name: my_progress_dsn

Description:

Host Name: localhost

Port Number: 8910

Database Name: oebpsdev

User ID: altova

5. To verify connectivity before saving the entered data, click **Test Connect**. To save, click **OK**.
6. The new data source now appears in the list of ODBC data sources. Click **Connect** to start the connection.

## 11.2.10.19 Sybase (JDBC)

This example illustrates how to connect to a Sybase database server via JDBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- Sybase *jConnect* component must be installed on your operating system (in this example, *jConnect 7.0* is used, installed as part of the *Sybase Adaptive Server Enterprise PC Client* installation).
- You have the following database connection details: host, port, database path, name, or alias, username, and password.

### Connection

1. Start the database connection wizard[961] and click **JDBC Connections** (*see JDBC Connection*[979] *for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH[981] of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.

4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the Sybase example*

| Field | Value |
|---|---|
| Classpaths | `C:\sybase\jConnect-7_0\classes\jconn4.jar` |
| Driver | *com.sybase.jdbc4.jdbc.SybDriver* |
| Database URL | `jdbc:sybase:Tds:hostName:port/databaseName` |

## 11.2.10.20 Teradata (JDBC)

This example illustrates how to connect to a Teradata database server via JDBC.

### Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: (i) the custom JVM path you may have set in application **Options**; ; (ii) the JVM path found in the Windows registry; (iii) the `JAVA_HOME` environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC driver (one or more `.jar` files that provide connectivity to the database) must be available on your operating system. In this example, Teradata JDBC Driver 16.20.00.02 is used. For more information, see https://downloads.teradata.com/download/connectivity/jdbc-driver.
- You have the following database connection details: host, port, database path, name, or alias, username, and password.

### Connection

1. Start the database connection wizard[961] and click **JDBC Connections** (*see JDBC Connection*[979] *for a screenshot of the dialog*).
2. In the *Classpaths* field, enter the path to the `.jar` file that provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of `.jar` file paths. If you have added the filepath to the CLASSPATH[981] of the system, you can leave this field empty.
3. In the *Driver* field, select the appropriate driver. Relevant drivers will be available only if a valid `.jar` file path is in the *Classpaths* field or in the `CLASSPATH` environment variable.
4. Enter the username and password for the database in the corresponding fields.
5. Enter the JDBC connection string in the *Database URL* field according to the pattern in the table below, replacing the highlighted values with the ones applicable to your database server.
6. Click **Connect**.

*Connection details of the Teradata example*

| Field | Value |
|---|---|
| Classpaths | `C:\jdbc\teradata\` |
| Driver | *com.teradata.jdbc.TeraDriver* |
| Database URL | `jdbc:teradata://databaseServerName` |

## 11.2.10.21 Teradata (ODBC)

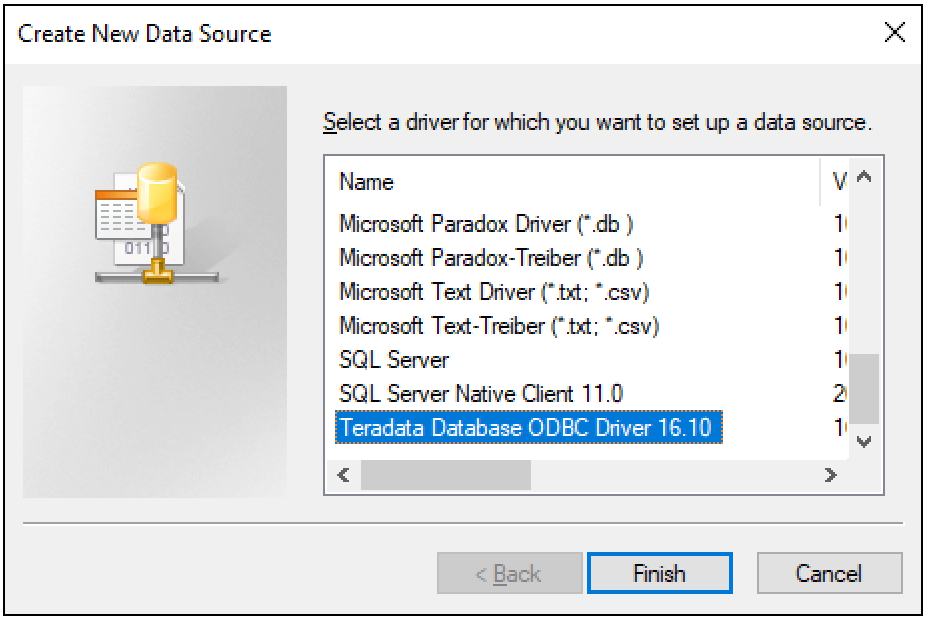
This example illustrates how to connect to a Teradata database server via ODBC.

### Prerequisites

- The Teradata ODBC driver must be installed (downloadable from the Teradata website).
- You have the following database connection details: host, username, and password.

### Connection

1. Press the **Windows** key, start typing `ODBC`, and select *Set up ODBC data sources (32-bit)* from the list of suggestions. (If appropriate, select the 64-bit option instead.)
2. In the ODBC Data Source Administrator dialog that appears, click the *System DSN* tab, and then click **Add**.
3. In the Create New Data Source dialog (screenshot below), select *Teradata Database ODBC Driver* and click **Finish**.

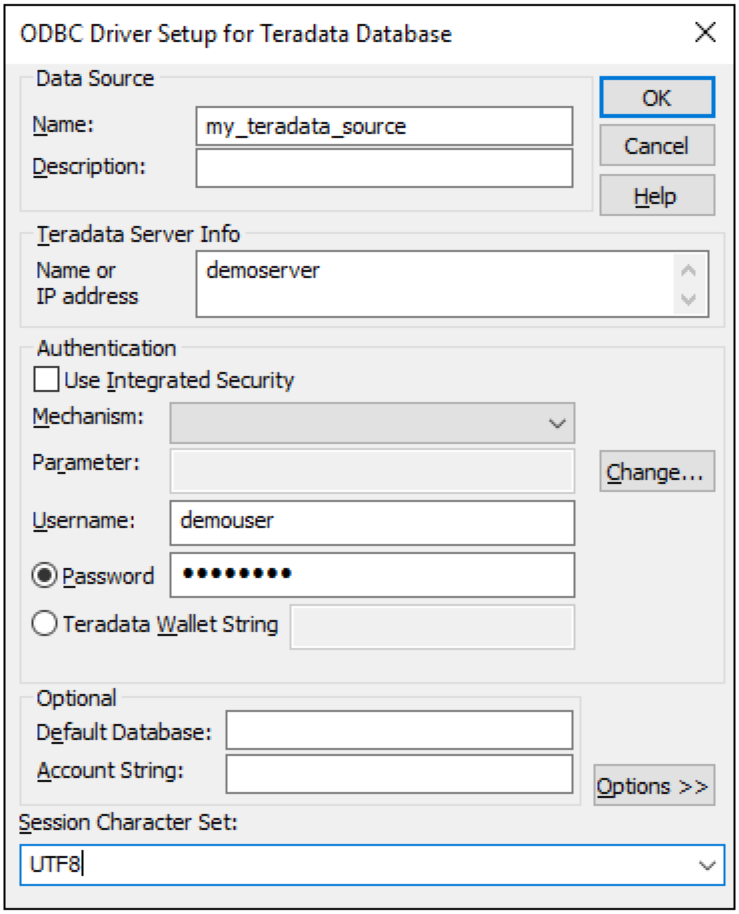


Create New Data Source

Select a driver for which you want to set up a data source.

| Name | V |
|---|---|
| Microsoft Paradox Driver (*.db ) | 1 |
| Microsoft Paradox-Treiber (*.db ) | 1 |
| Microsoft Text Driver (*.txt; *.csv) | 1 |
| Microsoft Text-Treiber (*.txt; *.csv) | 1 |
| SQL Server | 1 |
| SQL Server Native Client 11.0 | 2 |
| Teradata Database ODBC Driver 16.10 | 1 |

4. Enter the database and authentication details, and click **OK** when done.



ODBC Driver Setup for Teradata Database

Data Source

Name: my_teradata_source

Description:

Teradata Server Info

Name or IP address: demoserver

Authentication

□ Use Integrated Security

Mechanism:

Parameter:

Username: demouser

● Password ••••••••

○ Teradata Wallet String

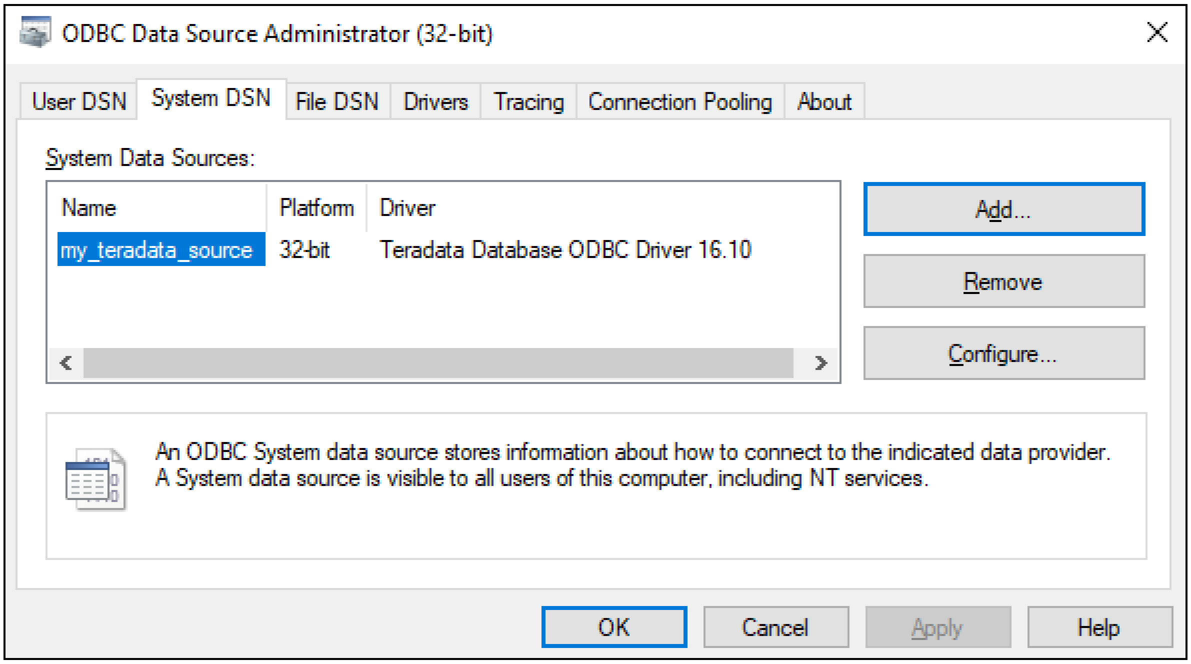
Optional

Default Database:

Account String:

Session Character Set: UTF8

5. Click **OK**. The data source now appears in the Data Sources list.



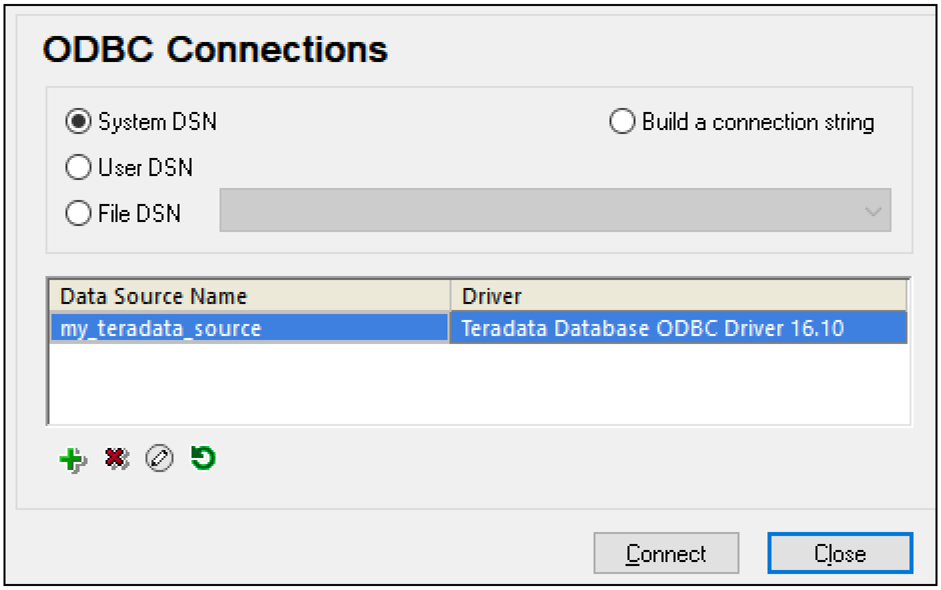
ODBC Data Source Administrator (32-bit)

System Data Sources:

| Name | Platform | Driver |
|---|---|---|
| my_teradata_source | 32-bit | Teradata Database ODBC Driver 16.10 |

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users of this computer, including NT services.

6. Run MobileTogether Designer and start the database connection wizard[961].
7. Click **ODBC Connections** and select *System DSN* (*see screenshot below*).



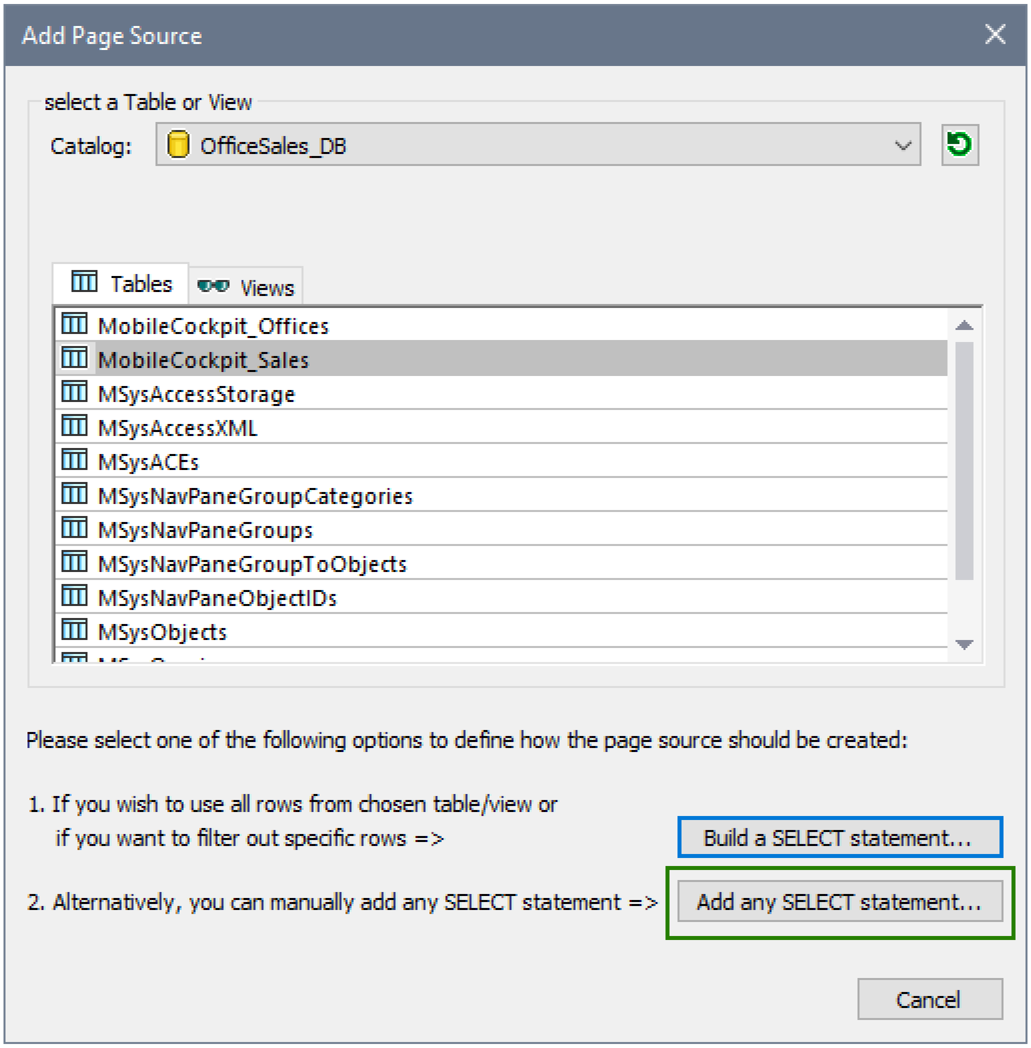
| Data Source Name | Driver |
|---|---|
| my_teradata_source | Teradata Database ODBC Driver 16.10 |

8. The data source you just created will be listed. Click **Connect** to start teh connection.

**Note:** If you get the following error: *"The driver returned invalid (or failed to return) SQL_DRIVER_ODBC_VER: 03.80"*, then make sure that the path to the ODBC client (for example, `C:\Program Files\Teradata\Client\16.10\bin`) exists in your system's **`PATH`** environment variable. If this path is missing, then add it manually.

## 11.3 Selecting DB Objects as Data Sources

After connecting to a DB, the Add Page Source dialog (*screenshot below*) is displayed. In this dialog, you can select the DB object (table or view) that you wish to add as a data source. All the columns of the table or view will be selected by default. You can refine your selection in the next steps by using the flexibility offered by SQL SELECT statement. You can either build a SELECT statement with the help of MobileTogether's DB wizard or you can build it manually. Click the appropriate button at the bottom right of the dialog. A SELECT statement can also serve to access other database object types, including, but not limited to, stored procedures.


Add Page Source

select a Table or View

Catalog: OfficeSales_DB

Tables | Views

- MobileCockpit_Offices
- MobileCockpit_Sales
- MSysAccessStorage
- MSysAccessXML
- MSysACEs
- MSysNavPaneGroupCategories
- MSysNavPaneGroups
- MSysNavPaneGroupToObjects
- MSysNavPaneObjectIDs
- MSysObjects

Please select one of the following options to define how the page source should be created:

1. If you wish to use all rows from chosen table/view or if you want to filter out specific rows => Build a SELECT statement...

2. Alternatively, you can manually add any SELECT statement => Add any SELECT statement...

Cancel

Note the following points:

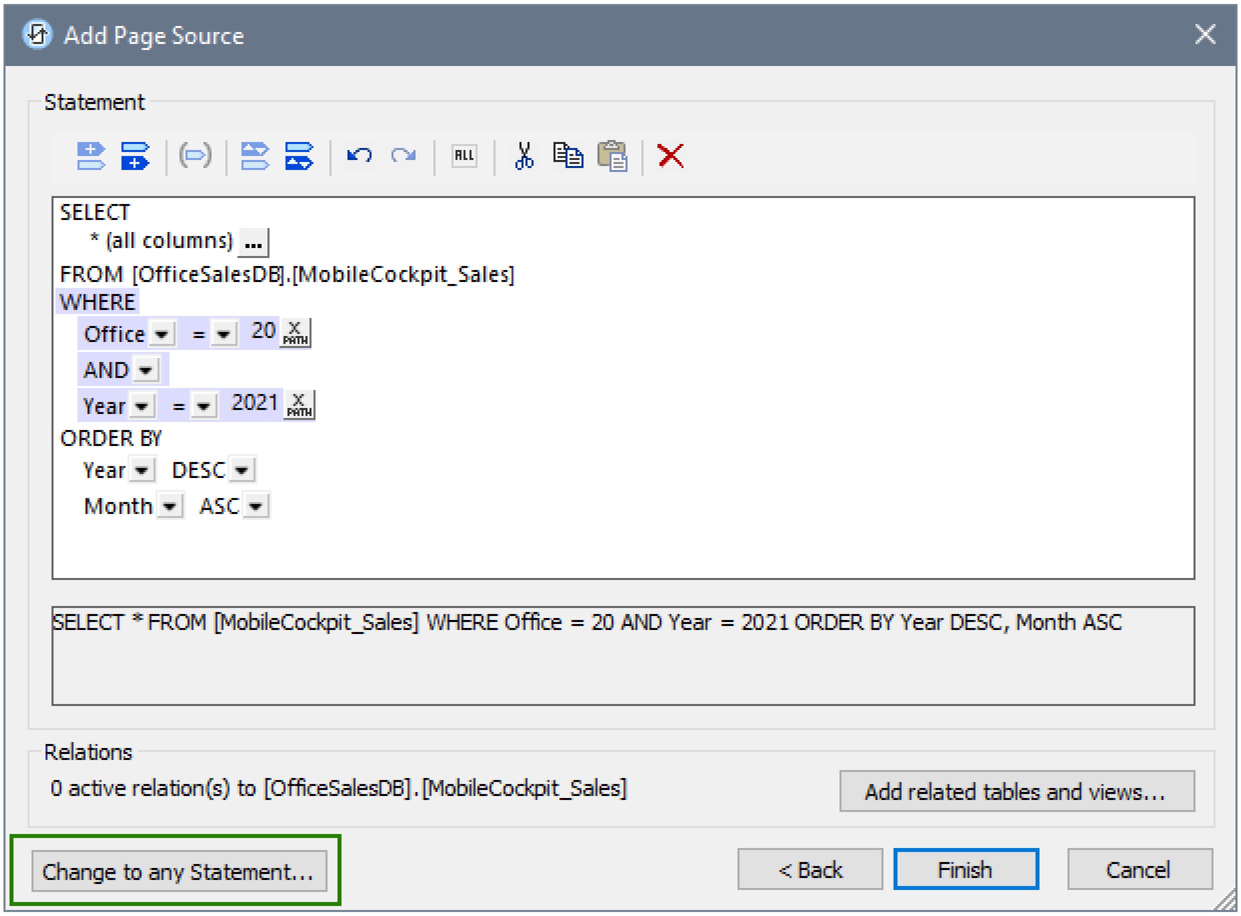
- The DB to which you connected is shown as the selected DB source. It cannot be changed.
- Only one table/view can be selected as a data source.
- You can filter in/out specific rows of the selected table/view by using a SELECT statement, as well as use other search capabilities of the SELECT statement.
- You can build a SELECT statement based on the currently selected table/view (double-click the selected table/view or click **Build a SELECT statement**). Alternatively, you can enter a SELECT

statement using SQL code (click **Add any SELECT statement**). The dialog that appears in each case is described below.

**Note:** This dialog (*screenshot above*) appears only when you add a data source for the first time. It enables you to select a data source. After a page source has been created, you can only edit the `SELECT` statement of the data source; you cannot change the data source.

## Build a SELECT statement

When, in the Add Page Source dialog (*screenshot above*), you double-click a table/view in the list box or click the **Build a SELECT statement** button, the dialog to build a `SELECT` statement appears (*screenshot below*). This dialog is aware of the currently selected table and provides you with context-sensitive interactive help to build a `SELECT` statement. You can use toolbar buttons and dialog controls (such as combo boxes and buttons) to build the statement, which will be displayed in a pane below the Build pane.



```
SELECT
    * (all columns) ...
FROM [OfficeSalesDB].[MobileCockpit_Sales]
WHERE
    Office = 20
    AND
    Year = 2021
ORDER BY
    Year DESC
    Month ASC
```

```
SELECT * FROM [MobileCockpit_Sales] WHERE Office = 20 AND Year = 2021 ORDER BY Year DESC, Month ASC
```

0 active relation(s) to [OfficeSalesDB].[MobileCockpit_Sales]

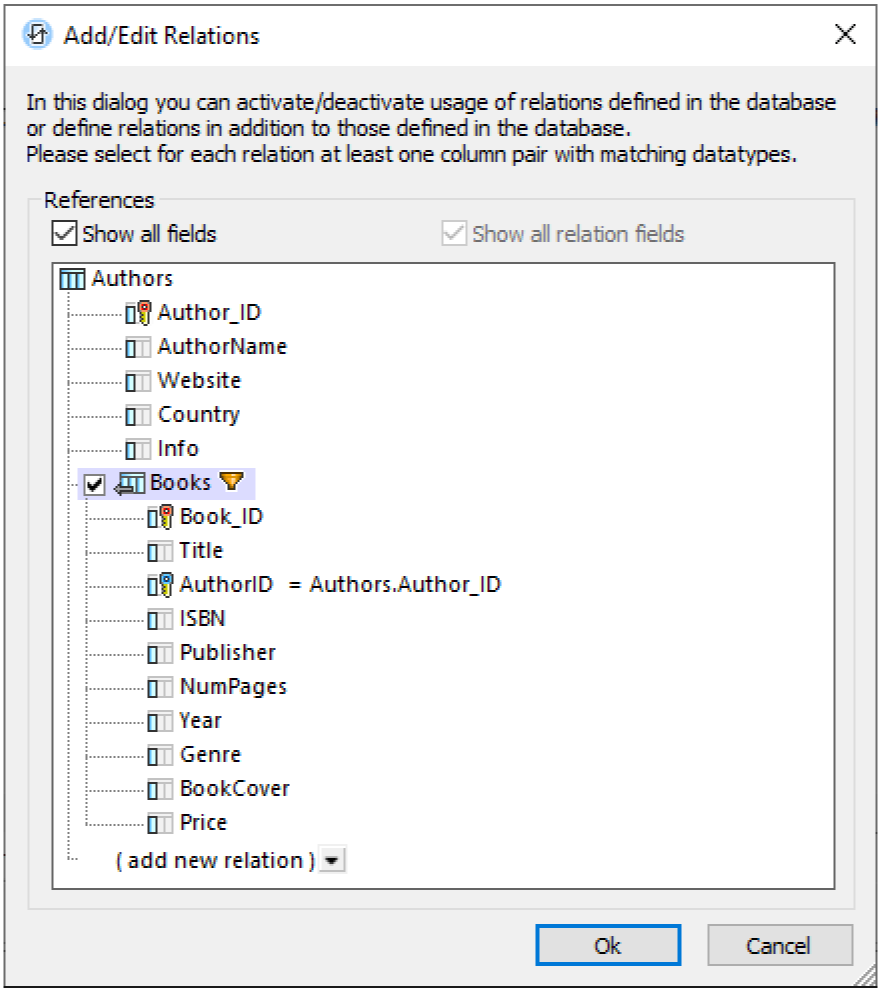
Use the following mechanisms to build a `SELECT` statement:

- For the `SELECT` keyword, choose the columns you want by clicking the **Additional Dialog** button. In the Columns dialog that appears, check the *Use all columns* check box to use the `SELECT *` instruction for selecting all columns, or check the individual columns you want to select.
- The value of the `FROM` keyword will be the name of the DB table to use. It is pre-selected and cannot be modified. If you want more flexibility in editing your `SELECT` statement, use the Add/Modify Any SELECT Statement dialog (*see below*).

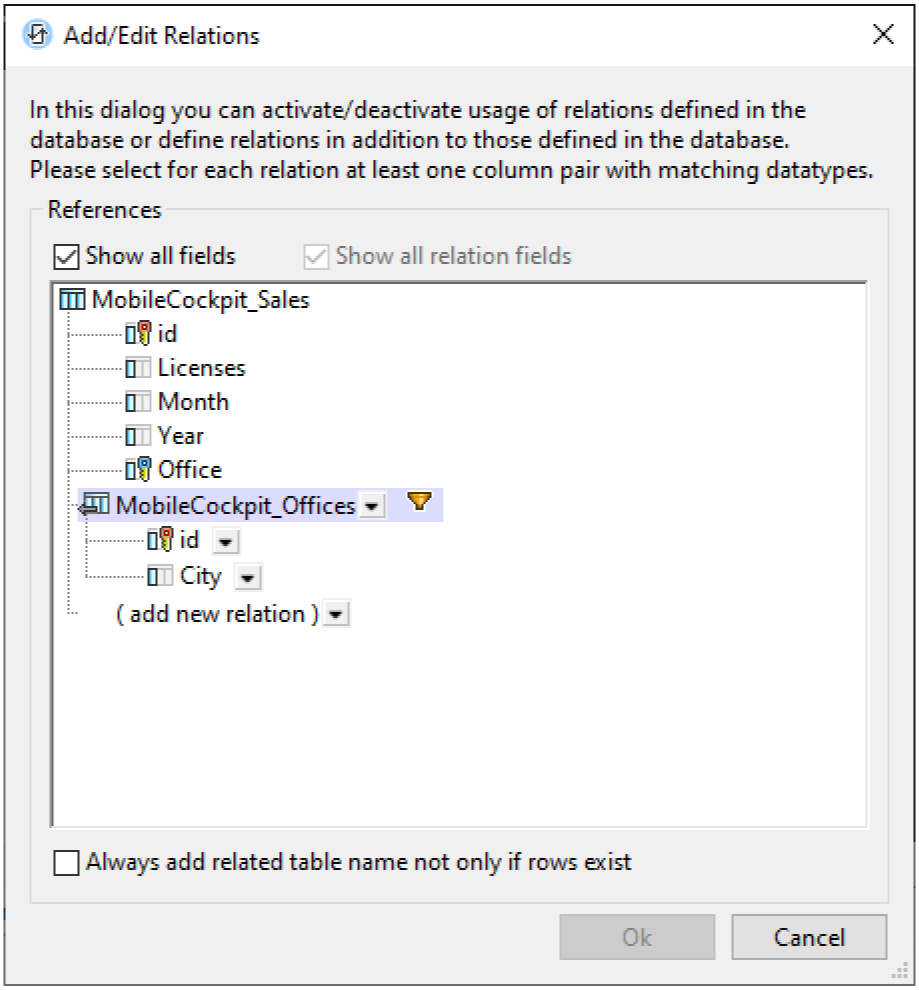
- Add expressions for the `WHERE` clause by using the **Insert expression before/behind** toolbar buttons. (To insert an expression before, you must select another expression.) For each expression, you can (i) choose one of the available columns in the first combo box, (ii) choose a **`WHERE`** clause conditional operator in the second combo box, and (iii) enter, as an XPath expression, a conditional value for the selected column .
- When you add a new expression to a `WHERE` clause, the logical `AND` operator will be pre-selected as the joining operator. You can change the logical operator (`AND` or `OR`) via the combo box.
- Create complex filters by grouping one or more expressions of the `WHERE` clause in parentheses (brackets). Select a set of adjacent expressions (including the logical operators within the set) and select the toolbar button to enclose the selected set of expressions in brackets.
- To sort the output, add an `ORDER BY` clause, in which you can choose: (i) the column on which to sort, and (ii) the sort order (ascending or descending). To sort on multiple columns, add more `ORDER BY` clauses before or after the first `ORDER BY` clause, as required.
- The following editing commands are available: *Undo, Redo, Select Entire WHERE Clause, Cut, Copy, Paste,* and *Delete*.
- If you want to add relations from the current table/view to one or more other tables/views, you can add the related tables/views and build relations in a graphical interface. Click **Add related tables/view** to open the Add/Edit Relations dialog (*described below*).

*Add related tables and views*

In the Add/Edit Relations dialog (*screenshot below*), you can (i) enable relations that exist in the DB (*screenshot below left*), and (ii) create relations between tables (*screenshot below right*). This also enables you to create hierarchical DB page sources.



In this dialog you can activate/deactivate usage of relations defined in the database or define relations in addition to those defined in the database.
Please select for each relation at least one column pair with matching datatypes.



In this dialog you can activate/deactivate usage of relations defined in the database or define relations in addition to those defined in the database.
Please select for each relation at least one column pair with matching datatypes.

☑ Show all fields ☑ Show all relation fields

□ Always add related table name not only if rows exist

If, in the DB, tables are related to the main table, then these are each displayed with a check box. You can select the tables for which you want to maintain the relation. (The relations are built using primary and foreign keys). For example, in the screenshot above left, the *Books* table has an **`AuthorID`** foreign key that relates each book record to an author in the *Authors* table. If the relation is maintained (by checking the *Books* table), then the page source will be created so that *Books* is hierarchically related to *Authors* as a child. For an example of a how a hierarchical DB is used, see the Hierarchical Database[118] tutorial.

If you want to add a new relation to a column (or field) that is in another table, do this as follows:

1. Click the combo box of the Add New Relation entry, and select the table in which the related field is located. In the screenshot above right, for example, the `MobileCockpit_Offices` table has been added.
2. Select *Show All Relation Fields* and *Show All Fields* to show all fields of all tables/views.
3. In the added table, click the dropdown arrow of the field for which you want to create a relation.
4. The dropdown list of the this added table's field will display the names of the original table's columns. Select the column you want to relate to the current field. The related column will be displayed to the right of the equal sign. In the screenshot above, for example, the `id` field of the added `Offices` table is related to the `id` field of the original `Sales` table.
5. Click **OK** to finish.

Note the following points:

- Checking the *Show All Relation Fields* option displays only those columns that have relations. Checking the *Show All Fields* option displays all fields of all tables/views. If both these options are unchecked, then no fields are displayed.
- Clicking the **Filter** icon near the name of the added table/view (*see screenshot*), opens the Build Select Statement dialog (*see above*) in which you can build a statement to filter the added table/view.
- To add related table names even if no rows exist in the table, select the corresponding check box at the bottom of the dialog (*see screenshot above right*).

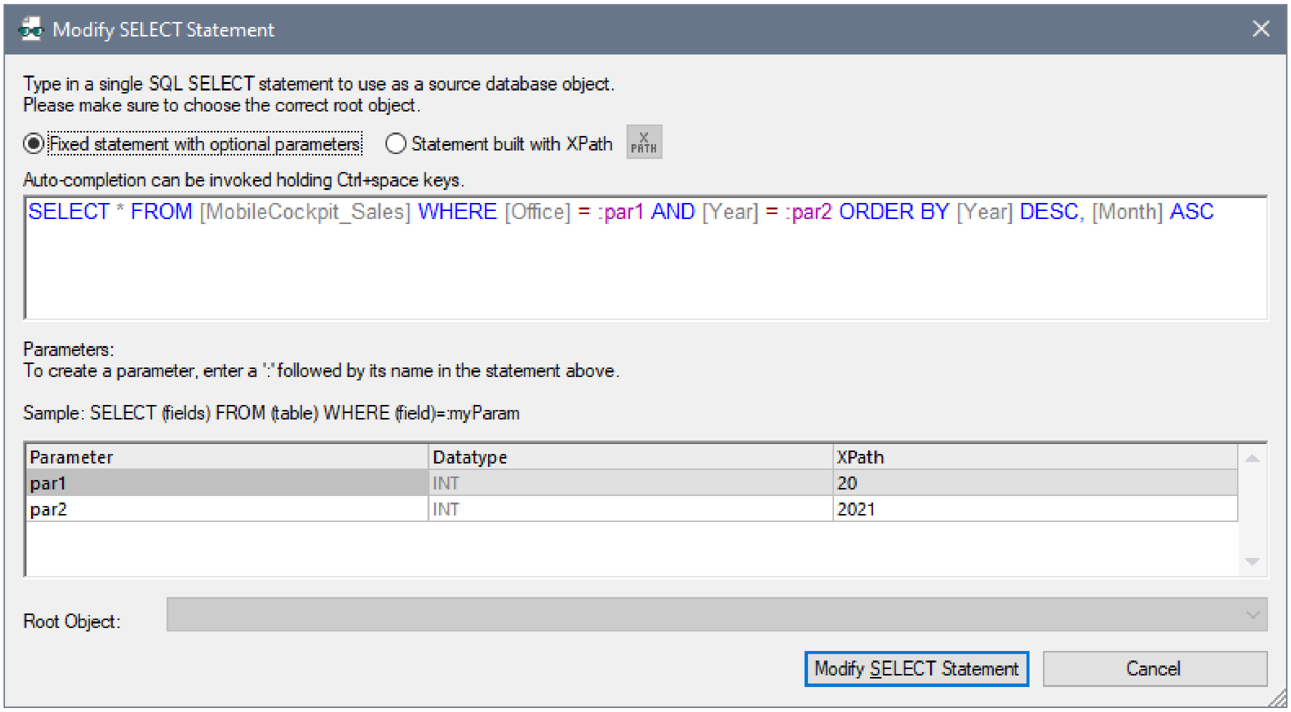
*Hierarchical DB-write support*
When the main data source is added as a page source of the design, its related tables (created as described above) will be displayed as child nodes of the main table. These child nodes can be assigned to controls of the design as usual and the hierarchical relations as shown in the page source tree can be used in XPath expressions in the design. Furthermore, if the main DB table is assigned to a Table control 614 and its child nodes are assigned to cells of the table control, then data is shown in the table according to the relations in the hierarchy. For an example of this, see the Hierarchical Database 118 tutorial.

For information about saving data to the DB, see Saving Data to the DB 1035.

## Add/modify a SELECT statement

The previous section describes the dialog to build a `SELECT` statement of the selected table/view. In that dialog, the table/view is pre-selected and cannot be changed. If, however, you want to create a `SELECT` statement without any restrictions or to modify an existing `SELECT` statement, you can use the Add/Modify SELECT Statement dialog (*screenshot below*). This dialog can be accessed at two points in the data selection procedure (see the buttons highlighted green in the two screenshots above).

- In the Select Table or View dialog (*first screenshot of this topic*), click **Add Any SELECT Statement**.
- In the Build a SELECT Statement dialog (*second screenshot of this topic*), click **Change to Any SELECT Statement**. The `SELECT` statement that you have created till this point will be displayed in the Modify SELECT Statement dialog (*screenshot below*), and you will be able to edit it.



Type in a single SQL SELECT statement to use as a source database object.
Please make sure to choose the correct root object.

● Fixed statement with optional parameters ○ Statement built with XPath

Auto-completion can be invoked holding Ctrl+space keys.

```
SELECT * FROM [MobileCockpit_Sales] WHERE [Office] = :par1 AND [Year] = :par2 ORDER BY [Year] DESC, [Month] ASC
```

Parameters:
To create a parameter, enter a ':' followed by its name in the statement above.

Sample: SELECT (fields) FROM (table) WHERE (field)=:myParam

| Parameter | Datatype | XPath |
|---|---|---|
| par1 | INT | 20 |
| par2 | INT | 2021 |

Root Object:

In the top pane, enter or modify the SELECT statement using the database query language SQL. Note from the screenshot above how you can use parameters instead of values. To use parameters, first enter a colon and a parameter name in the SELECT statement, for example, :par1. An entry for this parameter name will be created automatically in the *Parameters* pane in the bottom half of the dialog. Here, for each parameter, you can edit the parameter's datatype and supply a value for the parameter via an XPath expression.

After you finish editing your SELECT statement, click **Modify SELECT statement** to finish.

*Statement built with XPath*

You can also build an SQL statement with an XPath statement, which would typically be a string or a string concatenation. For example:

```
"SELECT * FROM Books WHERE ID = :iD"
concat("SELECT * FROM ", $XML1/MediaList/DBSelection, " WHERE ID = :iD")
```

After you finish editing your SELECT statement, click **Build a SELECT statement** or **Modify SELECT statement** to finish.

# 11.4 Editing DB Data

*This section*:

## About OriginalRowSet

In order for data to be edited and saved, the tree of the page source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. The original data is saved in the `OriginalRowSet` element, while edited data is saved in the `RowSet` element. When the page source is saved, the difference between the two trees, `OriginalRowSet` and `RowSet`, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to `OriginalRowSet` so that `OriginalRowSet` contains the newly saved DB data, and the modification process can be repeated.

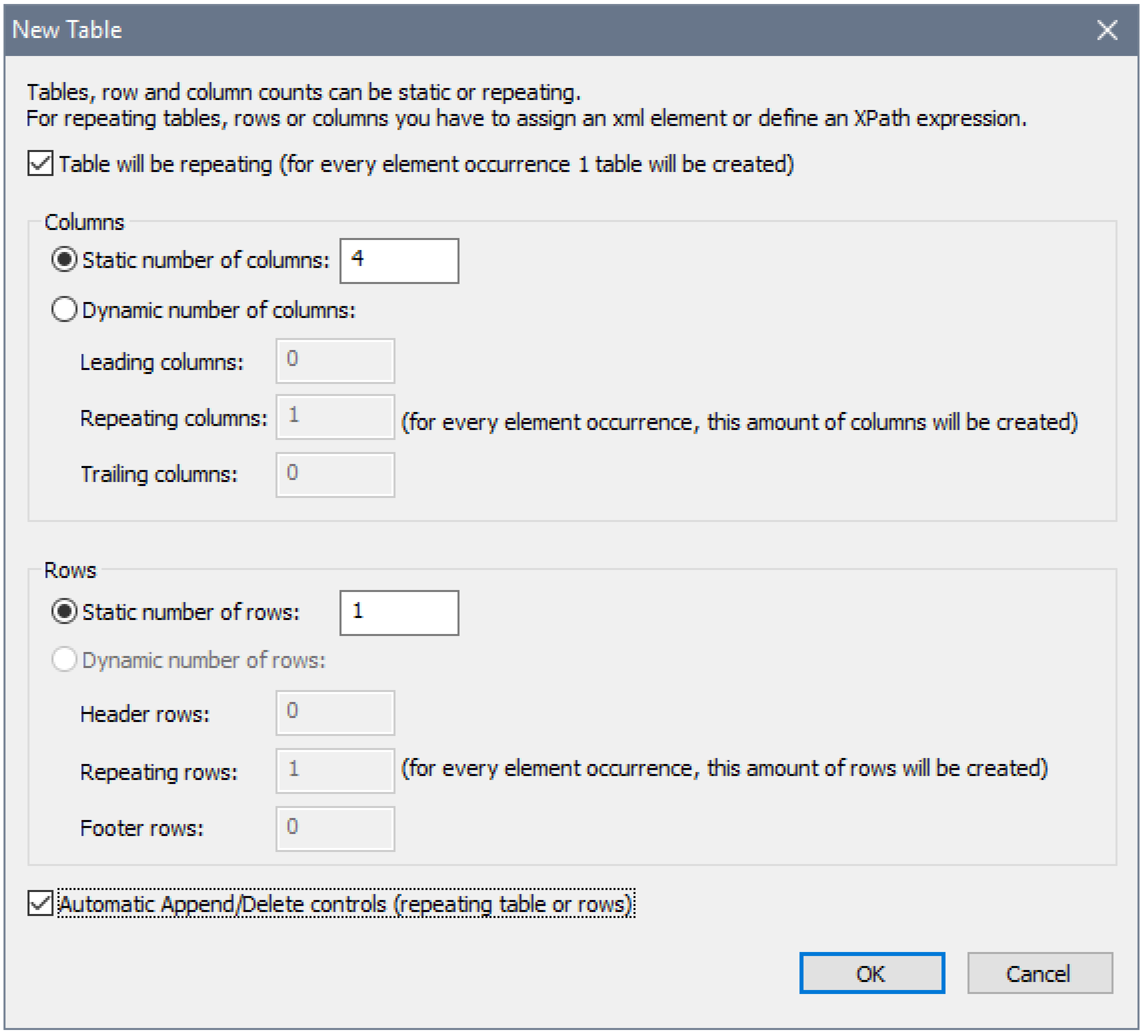
Note the following points:

- The `OriginalRowSet` element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command **Create OriginalRowSet**.
- The **Create OriginalRowSet**.command is enabled for database type (`$DB`) root nodes. It is a toggle command that creates/removes an `OriginalRowSet` data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the `OriginalRowSet` structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function **`mt-db-original-row`** 1262.

## Editing DB data that is in tables and in other controls

To create a control in which DB data can be edited, do the following:

- Use a control that is editable by the end user, such as a combo box or edit field. A label control, for instance, is not editable. If a table is used to generate repeating rows, add the editable controls within the cells of the table. See the sections Edit Offices Table 175 in the Database-And-Charts 159 tutorial for an example of how to do this. Also see the description of how to work with tables 1062.
- On the control, create a page source link to the page source node that is to be edited. Do this by dragging the page source node onto the control.
- If you use a table with repeating rows, use the option to automatically include Append/Delete controls when the table is created (*see screenshot below*).



Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

☑ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 4

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 1

○ Dynamic number of rows:

Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

☑ Automatic Append/Delete controls (repeating table or rows)

The advantage of using a table with repeating rows that are linked to the repeating `Row` elements of a DB page source is that when a table row is added, a DB row is also automatically added. For more information, see the description of how to work with tables [1062]. For examples, see the Database-And-Charts [159] tutorial and the MobileTogether Designer DB examples files.

## Updating, inserting, appending, and deleting nodes

The Update Data actions [873] enable nodes in DBs to be edited when a page or control event is triggered.

- Update Node(s) [886]: Updates one or more nodes, such as DB column, with value/s generated or obtained by the action's XPath expression.
- Insert Node(s) [880]: Inserts one or more nodes, such as DB rows, before a node selected by the action's XPath expression. The structure and contents of the inserted node can also be defined.
- Append Node(s) [875]: Appends one or more nodes, such as DB rows, as the first or last child of a node selected by the action's XPath expression. The structure and contents of the appended node can also be defined.
- Delete Node(s) [879]: Deletes one or more nodes, such as a DB rows, specified by the action's XPath expression.

**Note:** These actions are carried out on local data tree. The modified data tree must still be saved back to the DB[1035] for the end user modifications to be passed to the DB.

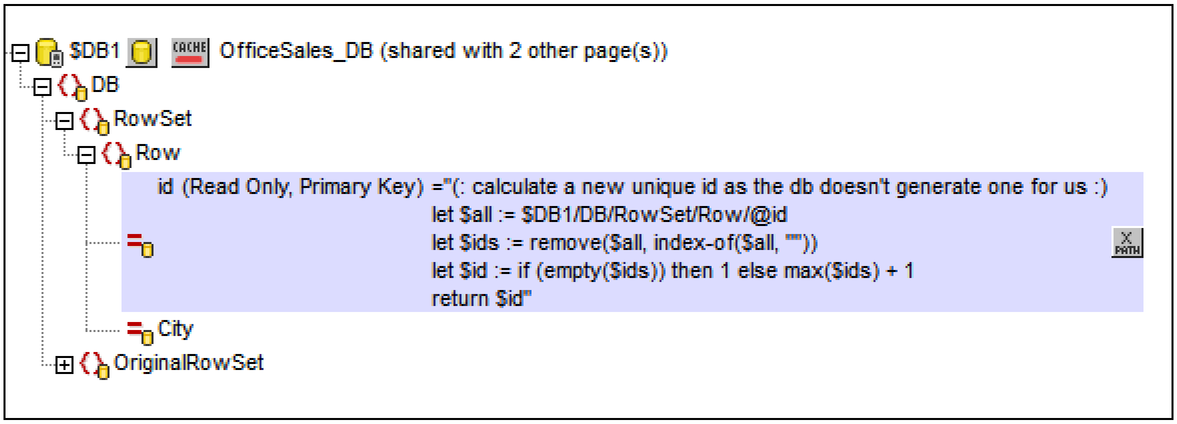
## The DB Execute action and `$MT_DBExecute_Result` variable

The DB Execute action[1040] enables you to use powerful SQL statements[861], including `INSERT`, `APPEND`, `UPDATE`, and `DELETE`, to modify a DB. It is different from the actions listed in the previous section[1032] in one important way: The modifications created by DB Execute's SQL statements are immediately saved to the DB. In the case of the actions listed in the previous section[1032] a Save mechanism[1035] must be used to save the modifications to the DB.

After the DB Execute action[861] executes an SQL statement, it stores the result in a variable named `$MT_DBExecute_Result`[1300]. This variable can then be used in XPath expressions anywhere within the project. Consequently, structures and data from one DB can be selected (optionally based on parameters)[861], then held in storage, modified, and inserted at other locations.

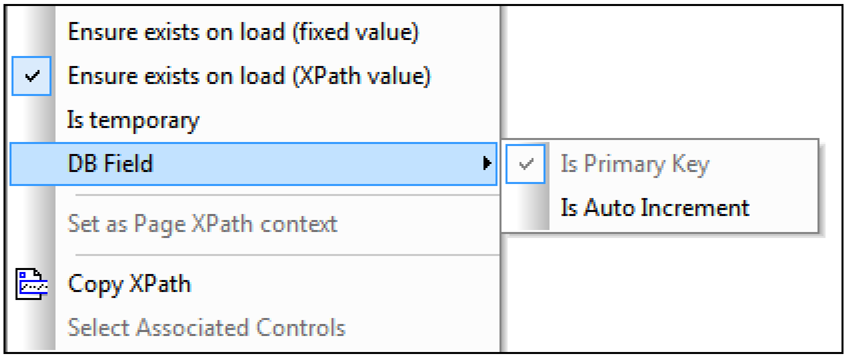
## Primary keys in MobileTogether Designer

Primary keys in DBs typically are auto-incrementing. When this is the case and a new row is added to a table, the primary key column of the added row is automatically incremented. In MobileTogether Designer, when a table is retrieved the primary key and auto-increment information is automatically retrieved and displayed in the Page Sources Pane (*see screenshot below*).



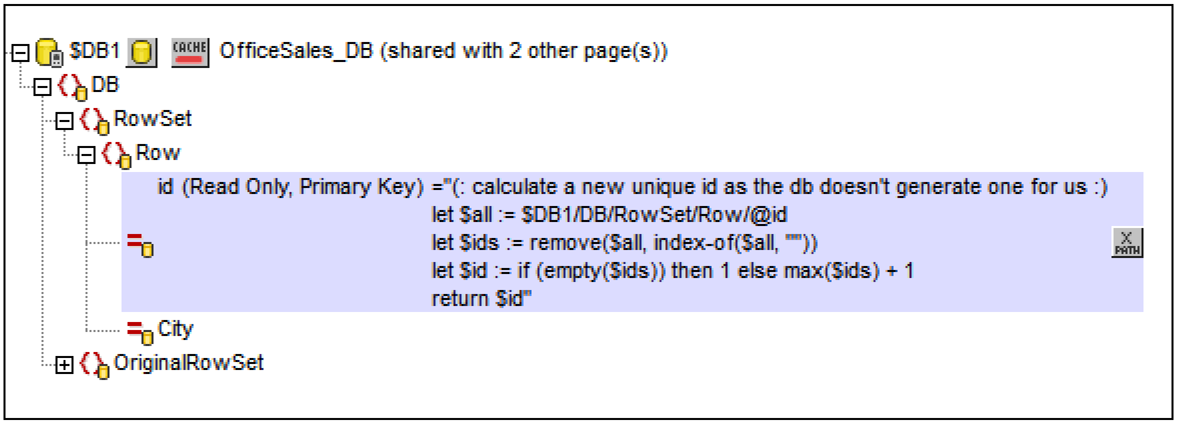
```
$DB1  OfficeSales_DB (shared with 2 other page(s))
  DB
    RowSet
      Row
        id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                                    let $all := $DB1/DB/RowSet/Row/@id
                                    let $ids := remove($all, index-of($all, ""))
                                    let $id := if (empty($ids)) then 1 else max($ids) + 1
                                    return $id"
        City
    OriginalRowSet
```

If auto-retrieval of this information was not successful, the context menu of tree nodes contains toggle commands that enable you to correctly annotate nodes (*see screenshot below*).


Ensure exists on load (fixed value)
✓ Ensure exists on load (XPath value)
Is temporary
DB Field
✓ Is Primary Key
Is Auto Increment
Set as Page XPath context
Copy XPath
Select Associated Controls

If the primary key column is not auto-incrementing, new primary key values for appended rows must be automatically generated using an XQuery expression. This is because primary key columns cannot be edited. The XQuery expression is inserted by using the primary-key node's context menu command, **Ensure Exists before Page Load (XPath Value)**. In the example below, a new value is generated for the primary key `@id` by using the following XQuery expression:

```
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```



```
id (Read Only, Primary Key) ="(: calculate a new unique id as the db doesn't generate one for us :)
                            let $all := $DB1/DB/RowSet/Row/@id
                            let $ids := remove($all, index-of($all, ""))
                            let $id := if (empty($ids)) then 1 else max($ids) + 1
                            return $id"
```

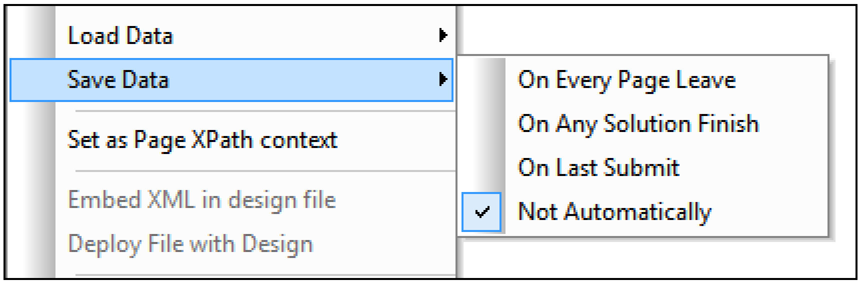
# 11.5 Saving Data to the DB

*This section*:

## Saving on the basis of solution progress

The context menu of a `$DB` root node has a **Save Data** command (*screenshot below*) which enables the data source represented by the root node to be updated at different points during the progress of the solution. The options are described below. If the default option, *Not Automatically*, is selected, then data is saved only when the *Save* action of an event is triggered.

The **Save Data** command rolls out a sub-menu with the following mutually exclusive options (only one can be selected):



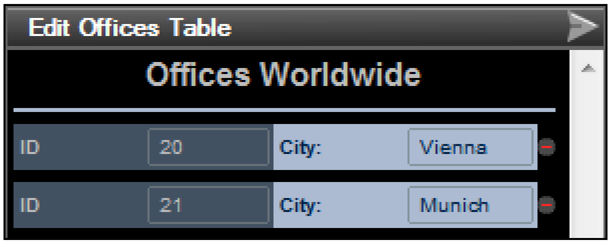
- *On Every Page Leave:* Data in the tree is saved every time a page containing that tree is exited.
- *On Any Solution Finish:* Data in the tree is saved when the solution is exited, no matter at what point or how the solution is exited.
- *On Last Submit:* Data in the tree is saved when the workflow progresses as designed, from first page to last page, and when the last **Submit** button is tapped. If this option is selected and the solution is exited before the last **Submit** button is tapped, then data in the tree will not be saved.
- *Not Automatically:* The tree will not automatically be saved. You must use the Save 801, Save to File 809, or Save to HTTP/FTP 827 actions to save data.

In the case of databases, there are options to save (i) modifications only; (ii) all rows; (iii) all rows if anything has been modified. If you wish to select an option that checks for modifications, then make sure that an **`OriginalRowSet`** 957 has been created for the table; otherwise an error will be reported on validation.

The default is *Not Automatically*.

## The Save action

Data can be saved to the DB when a page event or control event is triggered for which the Save[803] action has been defined. Such an event could be, for example, the clicking of the **Submit** button by the end user. In the screenshot below, the **Submit** button is located in the Edit Offices Table bar.

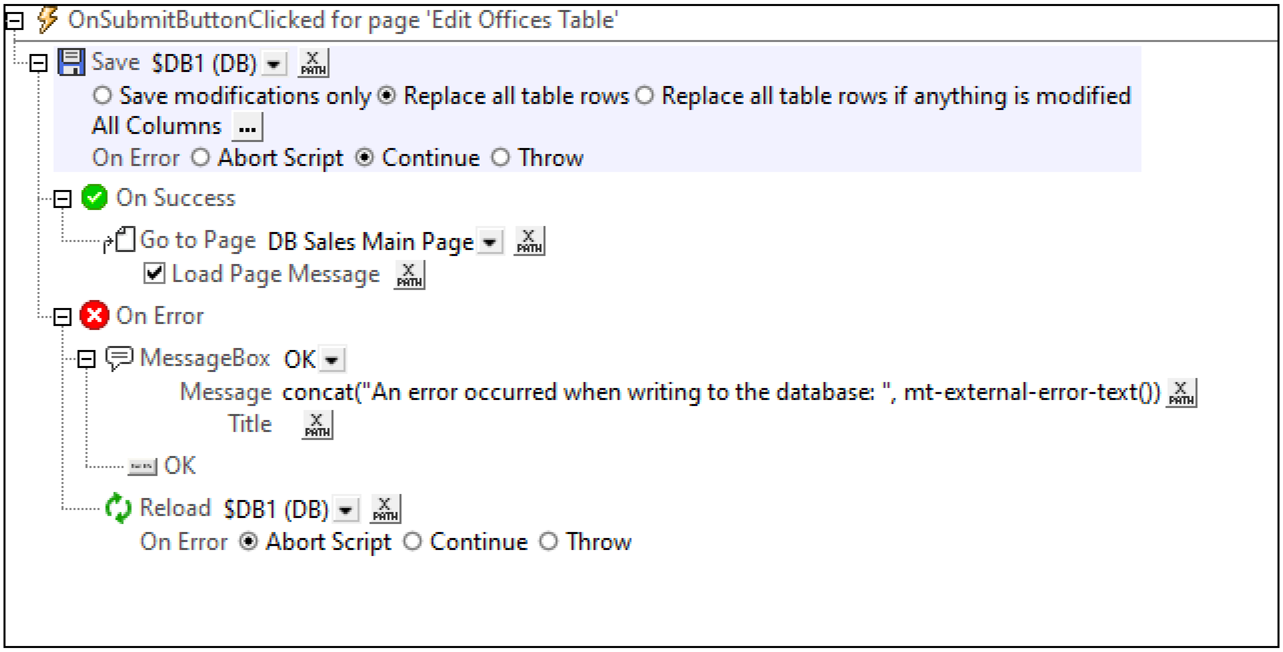


Edit Offices Table

Offices Worldwide

| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |

The *Save* action can be defined on a page action or control action. You can access the respective Actions dialogs via the All Actions dialog (**Page | Actions Overview**). The screenshot below shows a Page Actions dialog with the Save action defined for the `OnSubmitButtonClicked` event.



```
OnSubmitButtonClicked for page 'Edit Offices Table'
  Save  $DB1 (DB)
    ○ Save modifications only ● Replace all table rows ○ Replace all table rows if anything is modified
    All Columns ...
    On Error ○ Abort Script ● Continue ○ Throw
  On Success
    Go to Page  DB Sales Main Page
      ☑ Load Page Message
  On Error
    MessageBox  OK
        Message  concat("An error occurred when writing to the database: ", mt-external-error-text())
          Title
      OK
    Reload  $DB1 (DB)
      On Error ● Abort Script ○ Continue ○ Throw
```

**Note:** You can save (i) only those records that have been modified, added, or deleted; (ii) all records; (iii) all records if anything has been modified. If you wish to select an option that checks for modifications, then make sure that an `OriginalRowSet`[957] has been created for the table; otherwise an error will be reported on validation. Note that, if the DB has no private key, then the entire modified table will be saved to the DB, replacing the original table.

## The DB Execute action

The DB Execute action[861] is a powerful mechanism for modifying DB data. With this action, you can use SQL statements to update and save data. For information about using the action, see the section Page Design | Database | The DB Execute Action[1040].

## Saving related tables

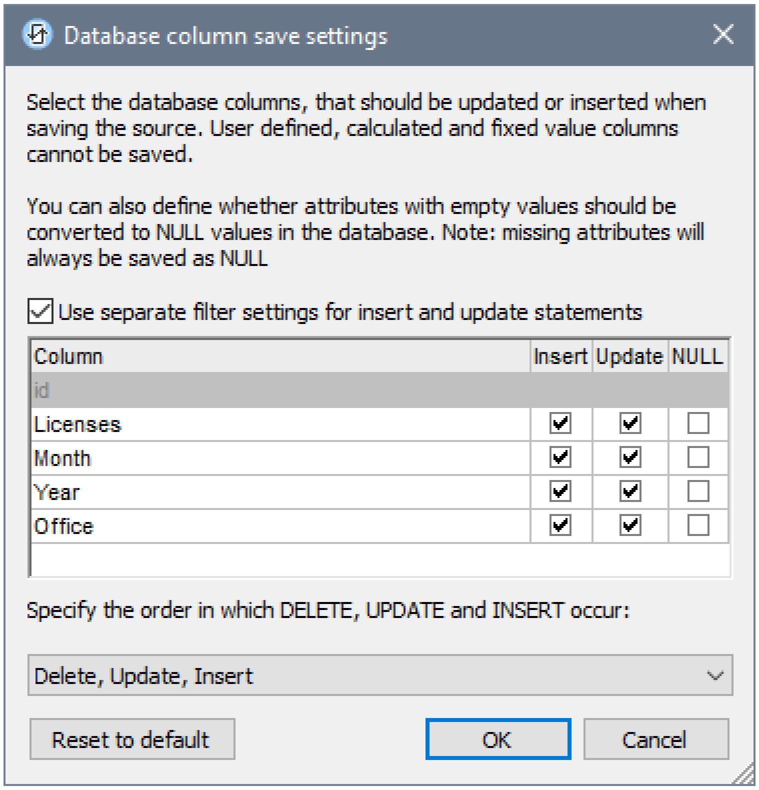
To specify how to save related tables, click the **Relations** button ... . This displays the Database Relation Save Settings dialog, which shows the related tables. In the combo box of each related table you can choose from the following options: (i) Replace all table rows of the related table; (ii) In the related table, save modifications only; (iii) Not save any changes to the related table.

When making these choices, take into account any private-key to foreign-key relations that exist between the main and related tables.

You can also access the save settings of the related table of a DB page source via the context menu of the page source[359] in the Page Sources Pane[270].

## Filtering the columns to save

In the context menu of `$DB` root nodes, select the command **Filter Columns** to display the Database Column Save Settings dialog (*screenshot below*) and to select which columns should be updated or inserted.



Database column save settings

Select the database columns, that should be updated or inserted when saving the source. User defined, calculated and fixed value columns cannot be saved.

You can also define whether attributes with empty values should be converted to NULL values in the database. Note: missing attributes will always be saved as NULL

☑ Use separate filter settings for insert and update statements

| Column | Insert | Update | NULL |
|---|---|---|---|
| id | | | |
| Licenses | ☑ | ☑ | □ |
| Month | ☑ | ☑ | □ |
| Year | ☑ | ☑ | □ |
| Office | ☑ | ☑ | □ |

Specify the order in which DELETE, UPDATE and INSERT occur:

Delete, Update, Insert

The dialog displays the columns of the DB page source. You can specify which columns can be updated or can take inserted values. (Updates refer to modified data in existing row elements; inserted values refer to data in newly added row elements.) By default, the *Insert* and *Update* options of each column are selected together as a pair. If, however, you wish to specify different options for a column's *Insert* and *Update* options, check the *Use separate filter settings for Insert and Update statements* check box. Attributes with empty values can be converted to `NULL` values in the DB by checking that column's *NULL* check box. Note that missing attributes will always be saved as `NULL`.

Columns that cannot be updated (because they are user-defined, fixed-value, or calculated-value) will not have an *Insert*, *Update*, or *NULL* option check box. In the screenshot above, the `ID` column cannot be updated because it holds fixed values. Deselect the columns you do not want to update.

You can specify the order in which deletions, updates, and insertions occur by selecting the desired order in the combo box at the bottom of the dialog.

If you wish to reset the *Save settings* so that all columns are updated, click **Reset to default**.

## About OriginalRowSet

In order for data to be edited and saved, the tree of the page source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. The original data is saved in the `OriginalRowSet` element, while edited data is saved in the `RowSet` element. When the page source is saved, the difference between the two trees, `OriginalRowSet` and `RowSet`, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to `OriginalRowSet` so that `OriginalRowSet` contains the newly saved DB data, and the modification process can be repeated.

Note the following points:

- The `OriginalRowSet` element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command **Create OriginalRowSet**.
- The **Create OriginalRowSet**.command is enabled for database type (`$DB`) root nodes. It is a toggle command that creates/removes an `OriginalRowSet` data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the `OriginalRowSet` structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function **`mt-db-original-row`**[1262].

## Committing transactions

Another way to save data to a DB is to begin an independent transaction and commit it. DB transactions are available as actions[854] for page and control events.

**About DB Transactions**
For each DB access that needs a transaction, one is automatically created and closed afterwards. This might not be desirable for some setups. For example, when you have two DB page sources that you want to update atomically together: If both tables are saved successfully, then the transaction is committed, but rolled back otherwise. To accommodate this kind of situation, transactions can be created on a connection basis.

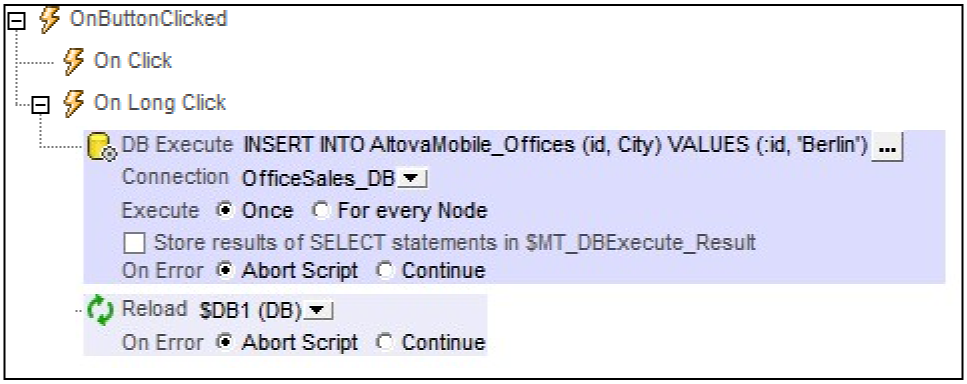
If you begin a transaction[856], all DB operations belonging to the same DB connection will use this transaction.

Committing a transaction[858] makes changes visible to the world outside your transaction. Changes can be rolled back[859]. In this case, even if you have done a Save on your page source, the changes won't be visible after a rollback! Note that any transaction that is not closed (committed or rolled back) when the end of the action tree has been reached will be rolled back automatically! A warning to this effect will be displayed in the Messages window.

It is important to bear in mind that, while the behavior above refers to explicit transaction actions, this behavior also applies to all DB operations that use the same connection as the transaction.

# 11.6 The DB Execute Action

The DB Execute action 861 (*see screenshot below*) is a powerful mechanism for modifying DB data. You can insert, delete, update, and save data by using SQL statements. This enables the power of the SQL language to be used whenever an event occurs during the progress of the solution.



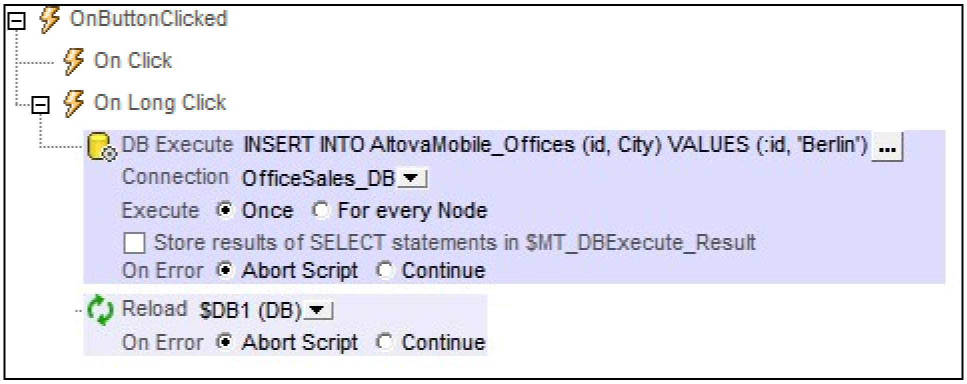
⊟ OnButtonClicked
On Click
⊟ On Long Click
DB Execute INSERT INTO AltovaMobile_Offices (id, City) VALUES (:id, 'Berlin') ...
Connection OfficeSales_DB
Execute ● Once ○ For every Node
□ Store results of SELECT statements in $MT_DBExecute_Result
On Error ● Abort Script ○ Continue
Reload $DB1 (DB)
On Error ● Abort Script ○ Continue

In this section, we describe how to insert, update, delete, and save data using DB Execute. The command to modify DB data is specified in the SQL statement of the action (*see screenshot above*). For a detailed description of the settings of the DB Execute action, see the section Actions > Database > DB Execute 861. Note that the SQL statement of DB Execute provides additional flexibility since it allows the use of parameters. The values of these parameters are generated by XPath expressions. See the DB Execute action 861 section for details.

If the DB data is being displayed on the same page as the page on which the action is defined, you should add a Reload action to update the display of the modified DB (*see screenshot above*). In the screenshot above, the `$DB1` tree is the root node of the database table `OfficeSales_DB`. After `OfficeSales_DB` has been modified with the `INSERT` statement, the `$DB1` tree on the design page is reloaded, thus immediately reflecting the modification to the DB.

## INSERT: Inserting rows with DB Execute and SQL

The `INSERT` statement of SQL can be used to insert rows in a database table. The `INSERT INTO` statement is used to insert rows with specific values, whereas the `INSERT SELECT` statement is used to insert the result of a `SELECT` statement into a table. You can also use other SQL statements, such as `SELECT INTO`, to insert rows in a table.



OnButtonClicked
On Click
On Long Click
DB Execute INSERT INTO AltovaMobile_Offices (id, City) VALUES (:id, 'Berlin')
Connection OfficeSales_DB
Execute ● Once ○ For every Node
□ Store results of SELECT statements in $MT_DBExecute_Result
On Error ● Abort Script ○ Continue
Reload $DB1 (DB)
On Error ● Abort Script ○ Continue

### *Insert a single complete row or a single partial row*

Use **INSERT INTO** to insert a single row into a table. The SQL syntax is:

```
INSERT INTO DestinationTable (ID, City) VALUES ('ID-Value', 'City-Value');
```

The statement above inserts a row containing two columns (`ID` and `City`) into the `DestinationTable` table. Note the following points:

- Only those columns that are specified in the SQL statement are inserted in the new row (`ID, City` in the example above).
- To insert a complete row (containing all table columns), specify all table columns in the SQL statement .
- The column names and column values in the SQL statement must correspond to each other by position. This column order does not need to correspond to the column order in the DB table. This means that if the layout of the DB table changes subsequently, the SQL statement will still be correct and does not need to be updated to reflect the changed layout.
- A column value must exist for each column name. Otherwise an error is generated and the row is not inserted.
- If a column is omitted in the SQL statement, then that column must be defined in the DB to allow `NULL` values (be empty) or to have a default value; otherwise an error is generated and the row is not inserted.
- To insert multiple rows, specify multiple `INSERT INTO` statements.

### *Insert the result of a SELECT statement*

Use **INSERT SELECT** to insert the result of a `SELECT` statement into a table. Typically, `INSERT SELECT` is used to copy a set of rows from one table to another. The SQL syntax is:

```
INSERT SELECT Offices (ID, City, Country) SELECT ('ID', 'Stadt', 'Land') FROM
Offices_DE ;
```

The statement above inserts all the rows of the `Offices_DE` table into the `Offices` table. Note the following points:

- Only those columns that are specified in the SQL statement are inserted in the new row (`ID, City, Country` in the example above).

- The columns returned by the SELECT statement will be inserted into the corresponding columns of the destination table. The correspondence of columns is determined by position. In the example above, for instance, the column Stadt at position 2 in the SELECT statement corresponds to the column City at position 2 in the definition of the destination table. The names of columns in the two definitions do not need to match; correspondence is fixed by position.
- The SELECT statement can use a WHERE clause to filter the data that is inserted.

## UPDATE: Updating rows with DB Execute and SQL

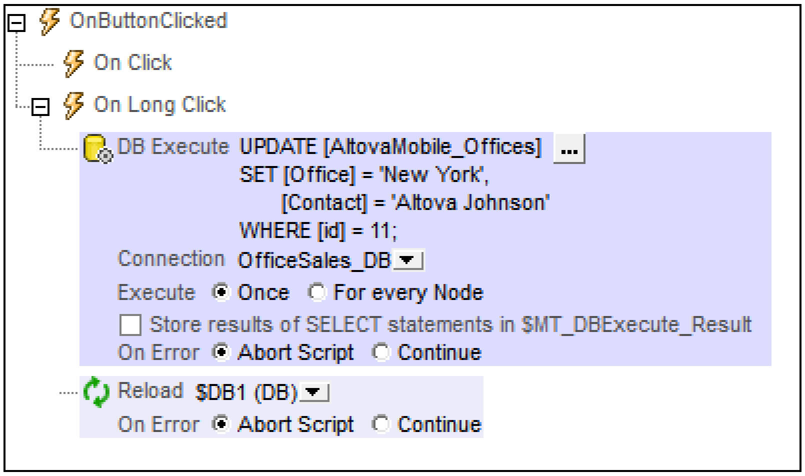
The UPDATE statement of SQL can be used to update rows in a database table. The UPDATE statement has three parts:

- The name of the DB table to update
- The names of the columns to update and their values
- A WHERE clause to filter which rows to update

Here is an example of an SQL UPDATE statement:

```
UPDATE [AltovaMobile_Offices]
SET    [Office]  = 'New York',
       [Contact] = 'Altova Johnson'
WHERE  [id]      = 11;
```

This statement updates the row with id=11 from, say, Office='USA' to Office='New York' and Contact=NULL to Contact='Altova Johnson'. The screenshot below shows this UPDATE statement example in the SQL statement setting of a DB Execute action 861.



```
OnButtonClicked
    On Click
    On Long Click
        DB Execute  UPDATE [AltovaMobile_Offices]
                    SET [Office] = 'New York',
                        [Contact] = 'Altova Johnson'
                    WHERE [id] = 11;
        Connection  OfficeSales_DB
        Execute  ● Once  ○ For every Node
        □ Store results of SELECT statements in $MT_DBExecute_Result
        On Error  ● Abort Script  ○ Continue
        Reload  $DB1 (DB)
        On Error  ● Abort Script  ○ Continue
```

Note the following points:

- The columns to be updated are given by their name=value combinations, with each name=value combination being separated from the next by a comma. There is no comma after the last name=value combination.
- All the columns to be updated are specified within a single SET clause.
- A column's value can be deleted by setting it to NULL, assuming that NULL values are allowed for that column. For example: SET [Contact] = NULL.

The Reload action reloads the modified DB immediately after the modification has been carried out. Without the Reload action, the modification will not be displayed on the page.

## DELETE: Deleting rows with DB Execute and SQL

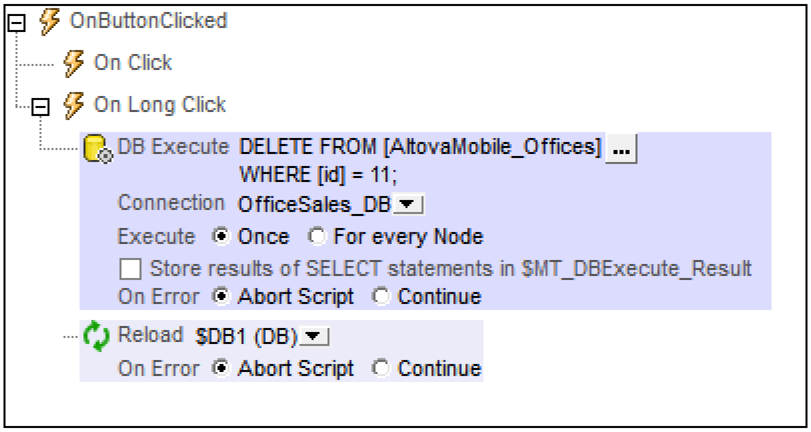
The `DELETE` statement of SQL can be used to delete:

- specific rows of a table (by specifying a `WHERE` clause to select the rows to delete)
- all rows of a table (by omitting the WHERE clause)

Here is an example of an SQL `DELETE` statement:

```
DELETE FROM [AltovaMobile_Offices]
WHERE [id] = 11;
```

The SQL `DELETE` statement above deletes the row with `id=11`. If the `WHERE` clause is omitted, then all the rows of the `AltovaMobile_Offices` table will be deleted.



OnButtonClicked
On Click
On Long Click
DB Execute DELETE FROM [AltovaMobile_Offices]
WHERE [id] = 11;
Connection OfficeSales_DB
Execute ● Once ○ For every Node
□ Store results of SELECT statements in $MT_DBExecute_Result
On Error ● Abort Script ○ Continue
Reload $DB1 (DB)
On Error ● Abort Script ○ Continue

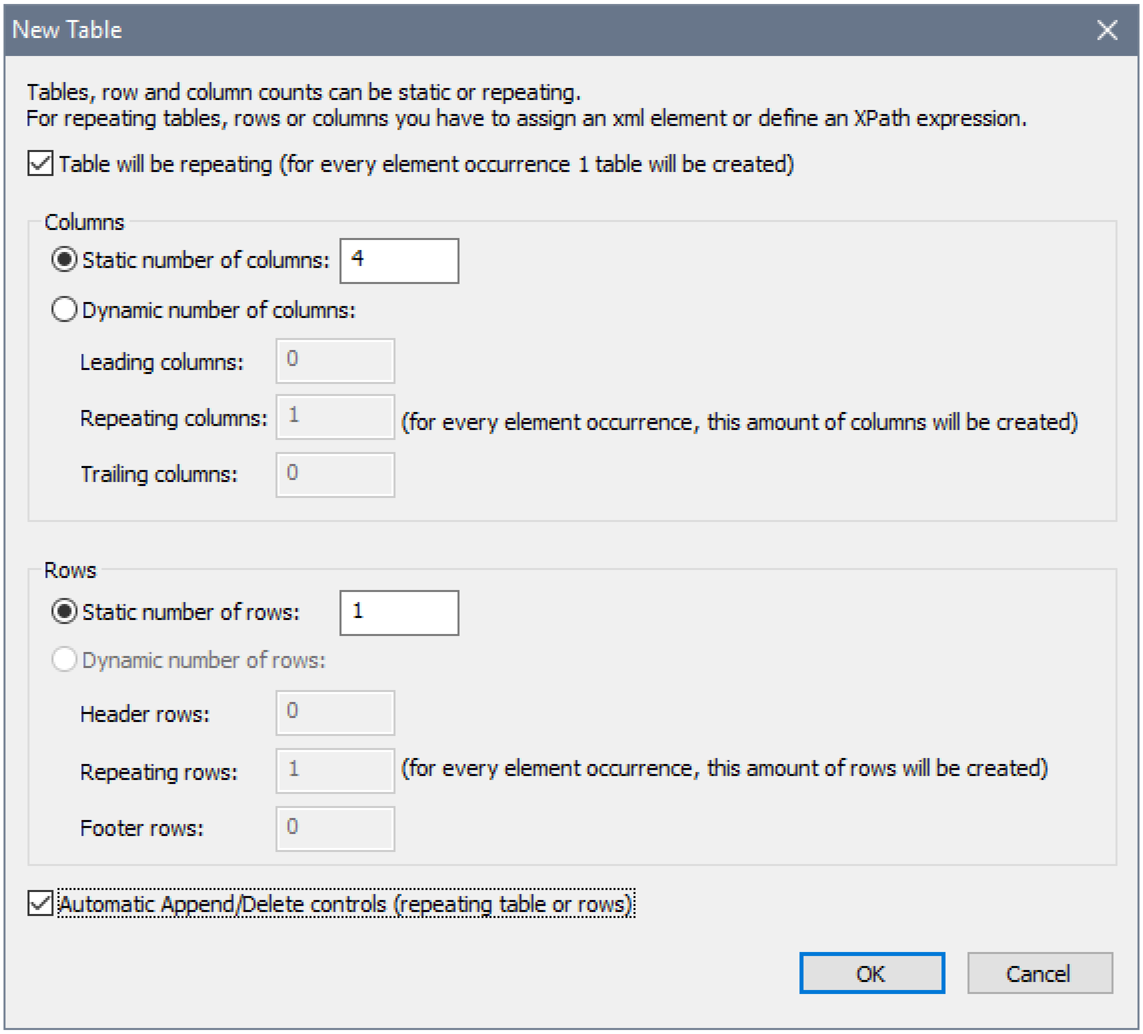
*An SQL DELETE statement in a DB Execute action.*

The Reload action reloads the modified DB immediately after the modification has been carried out. Without the Reload action, the modification will not be displayed on the page.

# 11.7 Displaying DB Data

## Displaying DB data in tables and other controls

DB data can be displayed in a control by creating a page source link from the control to a page source node. Typically the best way to display DB data is in a table with repeating rows. Drag a table control into the design and create a new table as a repeating table (*see screenshot below*). Then, drag controls into the cells of the table and make page source links to the column nodes of the DB row. For an actual demonstration of how this works, see the tutorial, Database-And-Charts[159].



Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

☑ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 4

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 1

○ Dynamic number of rows:

Header rows: 0

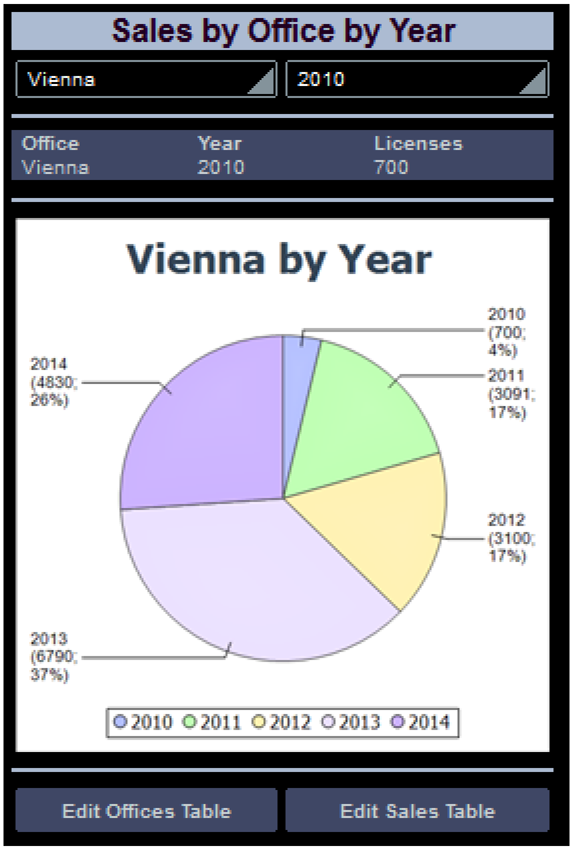
Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

☑ Automatic Append/Delete controls (repeating table or rows)

## Displaying DB data as charts

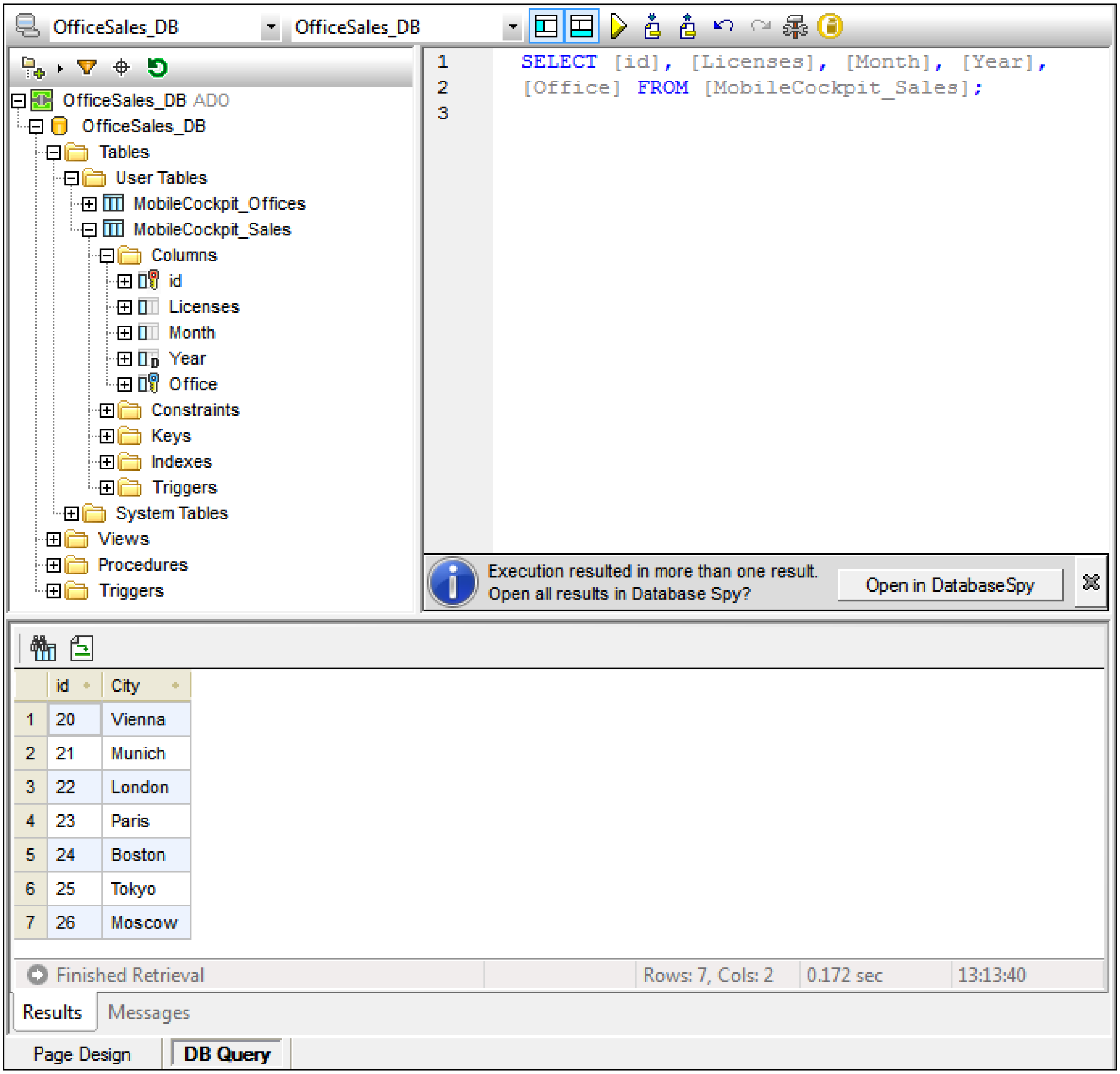
In addition to being able to display DB data directly, you can also create charts based on DB data.



For an example of how to do this, see the tutorial, Database-And-Charts 159.

# 11.8 Database Query

The DB Query View (*screenshot below*) enables you to directly query any major database from within the MobileTogether Designer GUI. The database could be a data source referenced in the active document or an external database. Note that each DB Query pane is associated with the currently active design. The DB Query pane of a design can have connections to multiple databases open at a time. There can also be multiple designs open at a time in MobileTogether Designer. Queries and actions defined in the DB Query View are independent of other MobileTogether Designer tabs, and are not saved as part of the design file.



```
SELECT [id], [Licenses], [Month], [Year],
[Office] FROM [MobileCockpit_Sales];
```

Execution resulted in more than one result.
Open all results in Database Spy?

|  | id | City |
|---|---|---|
| 1 | 20 | Vienna |
| 2 | 21 | Munich |
| 3 | 22 | London |
| 4 | 23 | Paris |
| 5 | 24 | Boston |
| 6 | 25 | Tokyo |
| 7 | 26 | Moscow |

Finished Retrieval    Rows: 7, Cols: 2    0.172 sec    13:13:40

## The Database Query mechanism

The Database Query mechanism is as follows. (It is described in detail in the sub-sections of this section.)

1. A connection to the database is established[1050] via the Database Query | Connect to a Data Source[1050] window.
2. The connected database or parts of it are displayed in the Browser pane[1052], which can be configured to suit viewing requirements.
3. A query[1055] written in a syntax appropriate to the database to be queried is entered in the Query pane[1055], and the query is executed.
4. The results of the query[1059] can be viewed through various filters.

## Supported databases

The table below lists all the supported databases. If your Altova application is a 64-bit version, ensure that you have access to the 64-bit database drivers needed for the specific database you are connecting to.

| Database | Notes |
|---|---|
| Firebird 2.x, 3.x, 4.x | |
| IBM DB2 8.x, 9.x, 10.x, 11.x, 12.x | |
| IBM Db2 for i 6.x, 7.4, 7.5 | Logical files are supported and shown as views. |
| IBM Informix 11.70 and later | |
| MariaDB 10 and later | MariaDB supports native connections. No separate drivers are required. |
| Microsoft Access 2003 and later | You can connect to an Access 2019 database from Altova products (i) only if the corresponding version of Microsoft Access Runtime is installed and (ii) only if the database does not use the "Large Number" data type. |
| Microsoft Azure SQL Database | SQL Server 2016 codebase |
| Microsoft SQL Server 2005 and later<br>Microsoft SQL Server on Linux | |
| MySQL 5 and later | MySQL 5.7 and later supports native connections. No separate drivers are required. |
| Oracle 9i and | |

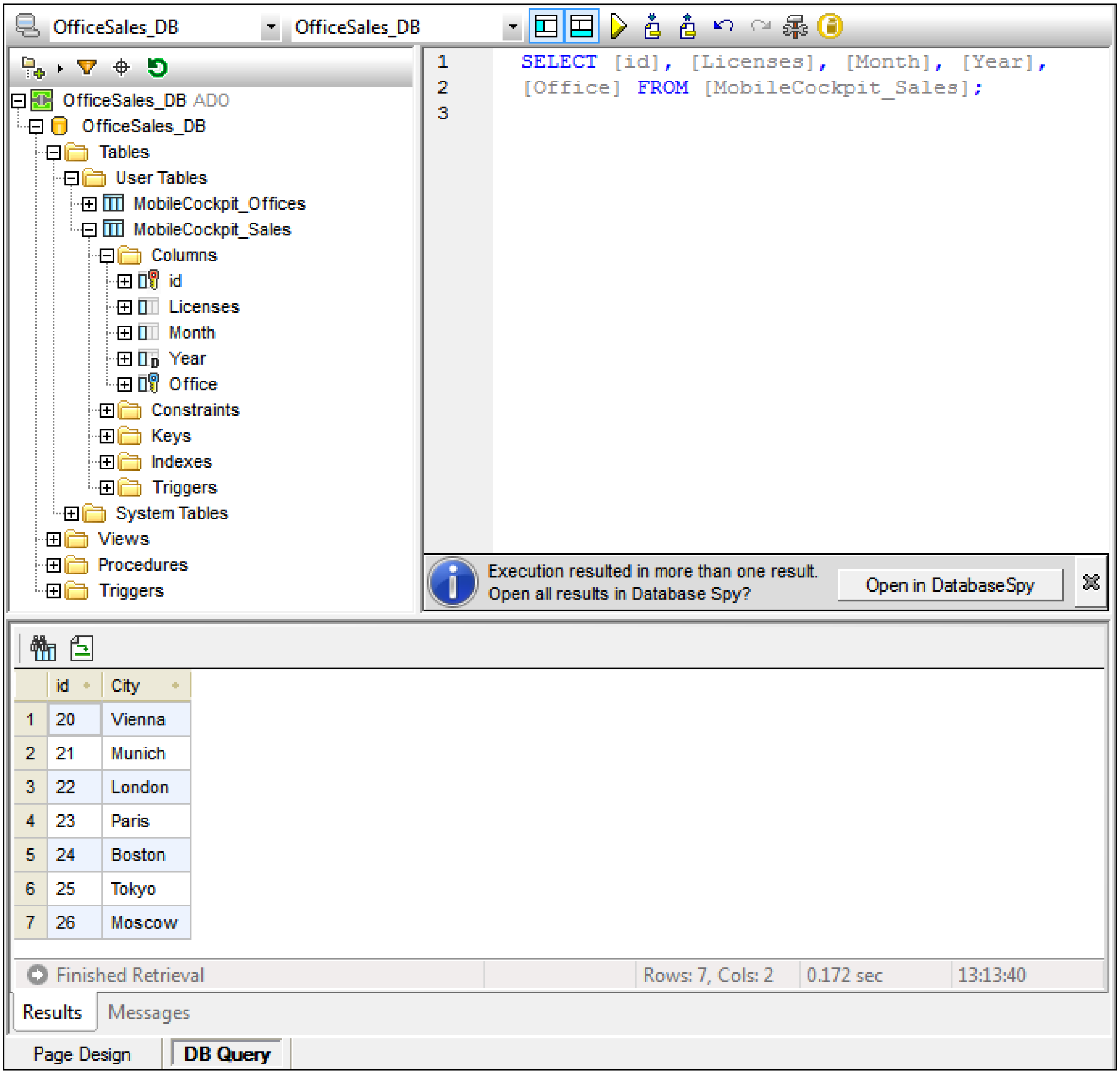
| Database | Notes |
|---|---|
| later | |
| PostgreSQL 8 and later | PostgreSQL connections are supported both as native connections and driver-based connections through interfaces (drivers) such as ODBC or JDBC. Native connections do not require any drivers. |
| Progress OpenEdge 11.6 | |
| SQLite 3.x | SQLite connections are supported as native, direct connections to the SQLite database file. No separate drivers are required. |
| Sybase ASE 15, 16 | |
| Teradata 16 | |

## 11.8.1 GUI Overview and Toolbar

The DB Query View (screenshot below) is divided into the following parts:

- The Browser window[1052] at left, which displays connection info and database tables
- The SQL Editor window (Query window)[1055], to the right of the Browser window, contains your SQL queries
- The Results tab of the Result/Messages windows[1059] displays the query results in tabular form
- The Messages tab of the Result/Messages windows[1059] displays warnings or error messages

These windows are described in detail in the sub-sections of this section.



```
SELECT [id], [Licenses], [Month], [Year],
[Office] FROM [MobileCockpit_Sales];
```

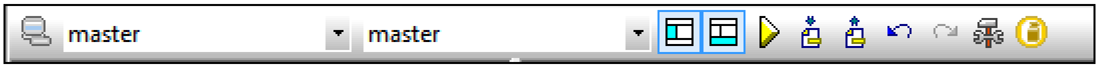
Execution resulted in more than one result. Open all results in Database Spy?

|  | id | City |
|---|---|---|
| 1 | 20 | Vienna |
| 2 | 21 | Munich |
| 3 | 22 | London |
| 4 | 23 | Paris |
| 5 | 24 | Boston |
| 6 | 25 | Tokyo |
| 7 | 26 | Moscow |

Finished Retrieval | Rows: 7, Cols: 2 | 0.172 sec | 13:13:40

## The DB Query View toolbar

The DB Query View toolbar (*screenshot below*) is located at the top of the view and provides icons for important commands related to the view.



| Icon | Command | Does this |
|---|---|---|

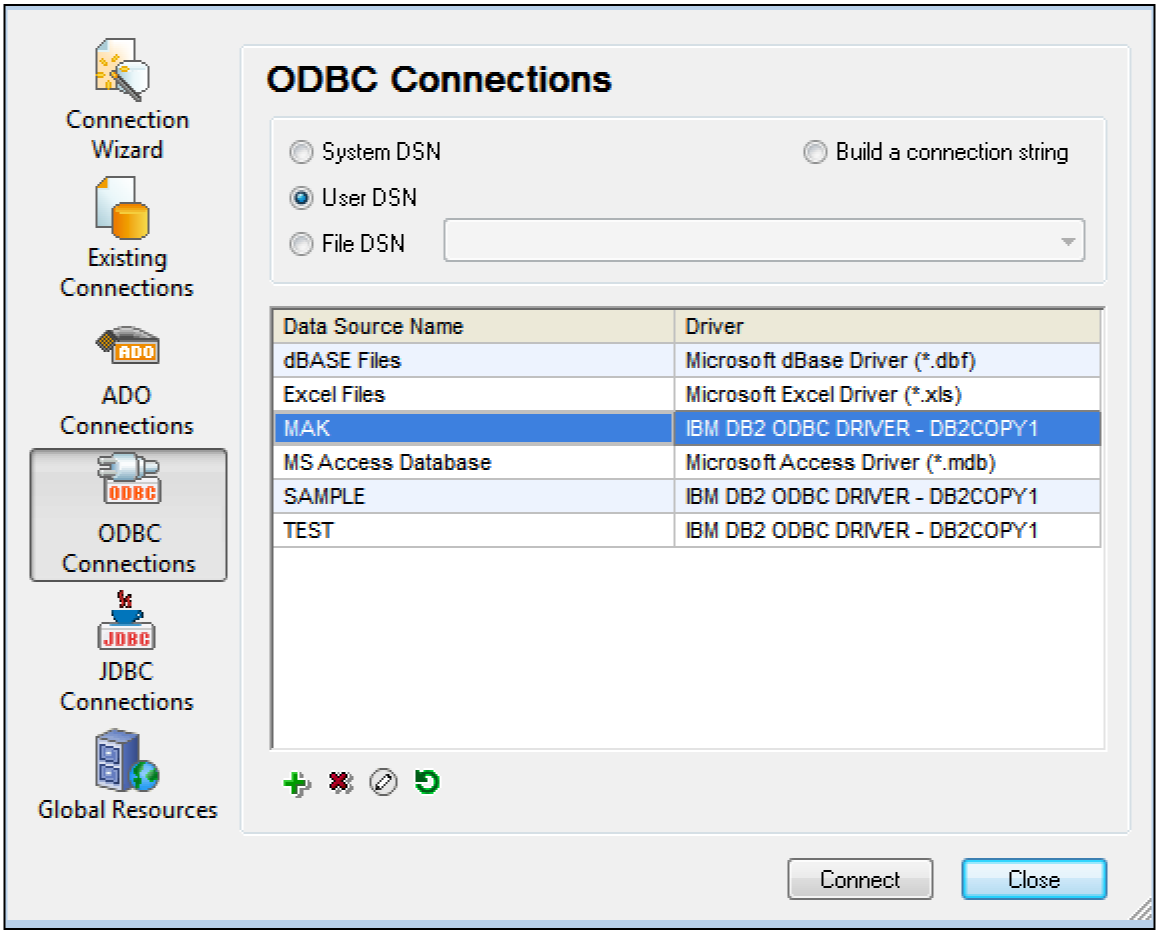
| | | |
|---|---|---|
| | **Quick Connect** | Starts the database connection wizard. |
| | **Toggle Browser** | Toggles the Browser window on and off. |
| | **Toggle Results** | Toggles the Results/Messages window on and off. |
| ▷ | **Execute Query** | Executes currently selected SQL statement. If script contains multiple statements and none is selected, then all are executed. |
| | **Import SQL File** | Opens an SQL file in the SQL Editor. |
| | **Export SQL File** | Saves SQL queries to an SQL file. |
| ↶ | **Undo** | Undoes an unlimited number of edits in SQL Editor. |
| ↷ | **Redo** | Redoes an unlimited number of edits in SQL Editor. |
| | **Options** | Open the Options dialog of SQL Editor. |
| | **Open Query in DatabaseSpy** | Opens the SQL script in Altova's DatabaseSpy application. |

## 11.8.2 Connecting to Data Sources

In order to query a database, you have to first connect to the required database, and then select the required data source and root object from among multiple existing connections.

### Connecting to a database

In the top left DB Query View, the click go to the any database that the active design uses as a data source is displayed in the Browser pane. If you wish to query some other database, click **Quick Connect** in the DB Query View toolbar.



## ODBC Connections

○ System DSN ○ Build a connection string

● User DSN

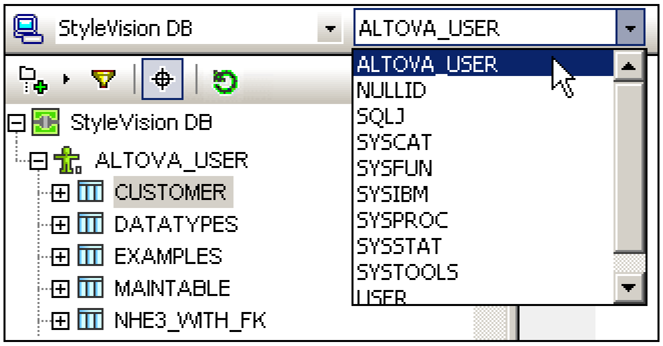
○ File DSN

| Data Source Name | Driver |
|---|---|
| dBASE Files | Microsoft dBase Driver (*.dbf) |
| Excel Files | Microsoft Excel Driver (*.xls) |
| MAK | IBM DB2 ODBC DRIVER - DB2COPY1 |
| MS Access Database | Microsoft Access Driver (*.mdb) |
| SAMPLE | IBM DB2 ODBC DRIVER - DB2COPY1 |
| TEST | IBM DB2 ODBC DRIVER - DB2COPY1 |

For a list of supported databases, see *Database Query | Supported databases*[1047].

## Selecting the required data source

All the existing connections and the root objects of each are listed, respectively, in two combo boxes in the toolbar of the Database Query window (*screenshot below*). After selecting the required data source in the left-hand combo box, you can select the required root object from the right-hand combo box.



In the screenshot above, the database with the name `StyleVision DB` has been selected. Of the available root objects for this database, the root object `ALTOVA_USER` has been selected. The database and the root object are then displayed in the Browser pane 1052.

## 11.8.3 Browser Pane

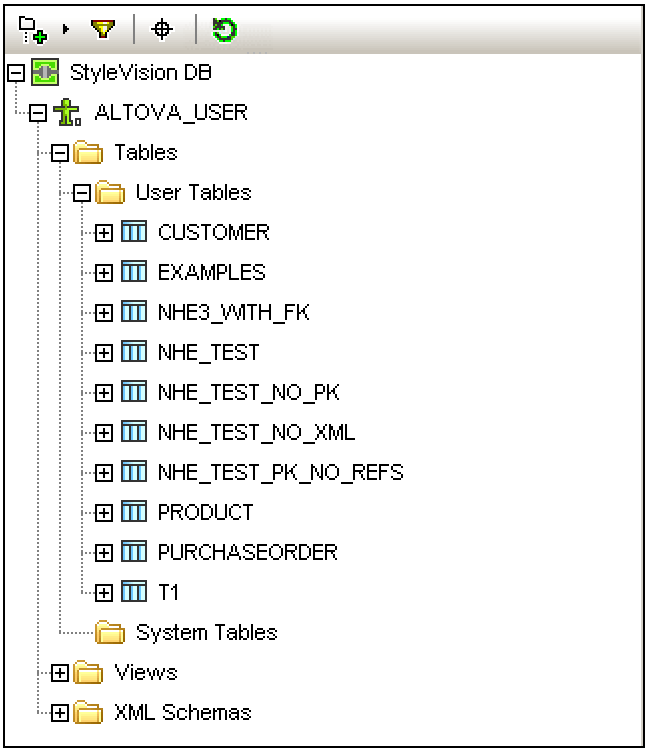
The Browser pane provides an overview of objects in the selected database. This overview includes database constraint information, such as whether a column is a primary or foreign key. In IBM DB2 version 9 and higher databases, the Browser additionally shows registered XML schemas in a separate folder (*see screenshot below*).

This section describes the following:

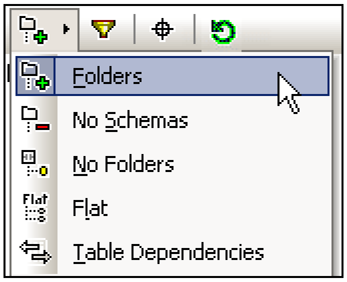
- The layouts 1052 available in the Browser pane.
- How to filter 1053 database objects.
- How to find 1054 database objects.
- How to refresh 1055 the root object of the active data source.

### Browser pane layouts

The default Folders layout displays database objects hierarchically. Depending on the selected object, different context menu options are available when you right-click an item.



To select a layout for the Browser, click the **Layout** icon in the toolbar of the Browser pane (*screenshot below*) and select the layout from the drop-down list. Note that the icon changes with the selected layout.



The available layouts are:

- *Folders:* Organizes database objects into folders based on object type in a hierarchical tree, this is the default setting.
- *No Schemas:* Similar to the Folders layout, except that there are no database schema folders; tables are therefore not categorized by database schema.
- *No Folders:* Displays database objects in a hierarchy without using folders.
- *Flat:* Divides database objects by type in the first hierarchical level. For example, instead of columns being contained in the corresponding table, all columns are displayed in a separate Columns folder.
- *Table Dependencies:* Categorizes tables according to their relationships with other tables. There are categories for tables with foreign keys, tables referenced by foreign keys and tables that have no relationships to other tables.

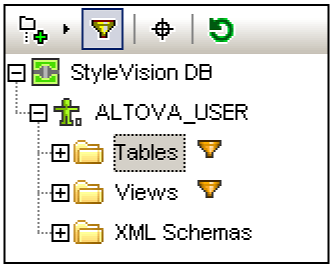
To sort tables into User and System tables, switch to Folders, No Schemas or Flat layout, then right-click the Tables folder and select **Sort into User and System Tables**. The tables are sorted alphabetically in the User Tables and System Tables folders.

## Filtering database objects

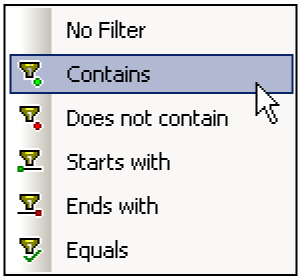
In the Browser pane (in all layouts except No Folders and Table Dependencies), schemas, tables, and views can be filtered by name or part of a name. Objects are filtered as you type in the characters, and filtering is case-insensitive by default.

To filter objects in the Browser, do the following:

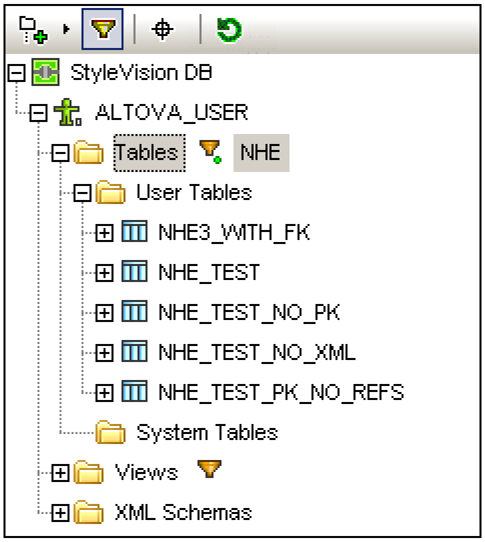
1. Click the Filter Folder Contents icon in the toolbar of the Browser pane. Filter icons appear next to the Tables and Views folders in the currently selected layout (*screenshot below*).



2. Click the filter icon next to the folder you want to filter and select the filtering option from the popup menu, for example, *Contains*.



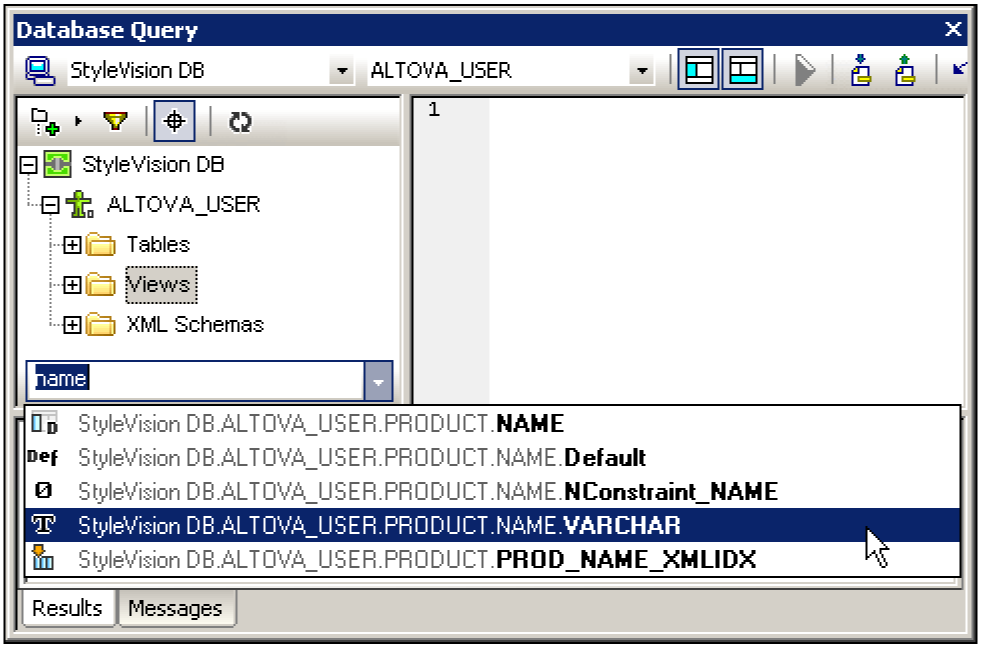
3. In the entry field that appears, enter the filter string (in the screenshot below, the filter string on the Tables folder is `NHE`). The filter is applied as you type.



## Finding database objects

To find a specific database item by its name, you can use the Browser pane's Object Locator. This works as follows:

1. In the toolbar of the Browser pane, click the Object Locator icon. A drop-down list appears at the bottom of the Browser pane.
2. Enter the search string in the entry field of this list, for example name (screenshot below). Clicking the drop-down arrow displays all objects that contain the search string.


Database Query
StyleVision DB
ALTOVA_USER
1
StyleVision DB
ALTOVA_USER
Tables
Views
XML Schemas
name
StyleVision DB.ALTOVA_USER.PRODUCT.NAME
StyleVision DB.ALTOVA_USER.PRODUCT.NAME.Default
StyleVision DB.ALTOVA_USER.PRODUCT.NAME.NConstraint_NAME
StyleVision DB.ALTOVA_USER.PRODUCT.NAME.VARCHAR
StyleVision DB.ALTOVA_USER.PRODUCT.PROD_NAME_XMLIDX
Results
Messages

3. Click the object in the list to see it in the Browser pane.

### Refreshing the root object

The root object of the active data source can be refreshed by clicking the **Refresh** button of the Browser pane's toolbar.

## 11.8.4 Query Pane: Description

The Query pane is an intelligent SQL editor for entering queries to the selected database. After entering the query, clicking the Execute command 1055 of the Database Query window executes the query and displays the result and execution messages in the Results/Messages pane 1059. How to work with queries is described in the next section, Query Pane: Working with Queries 1058. In this section, we describe the main features of the Query pane:

- SQL Editor icons 1055 in the Database Query toolbar
- SQL Editor options 1056
- Definition of regions in an SQL script 1057
- Insertion of comments in an SQL script 1057
- Use of bookmarks 1058

### SQL Editor icons in the Database Query toolbar

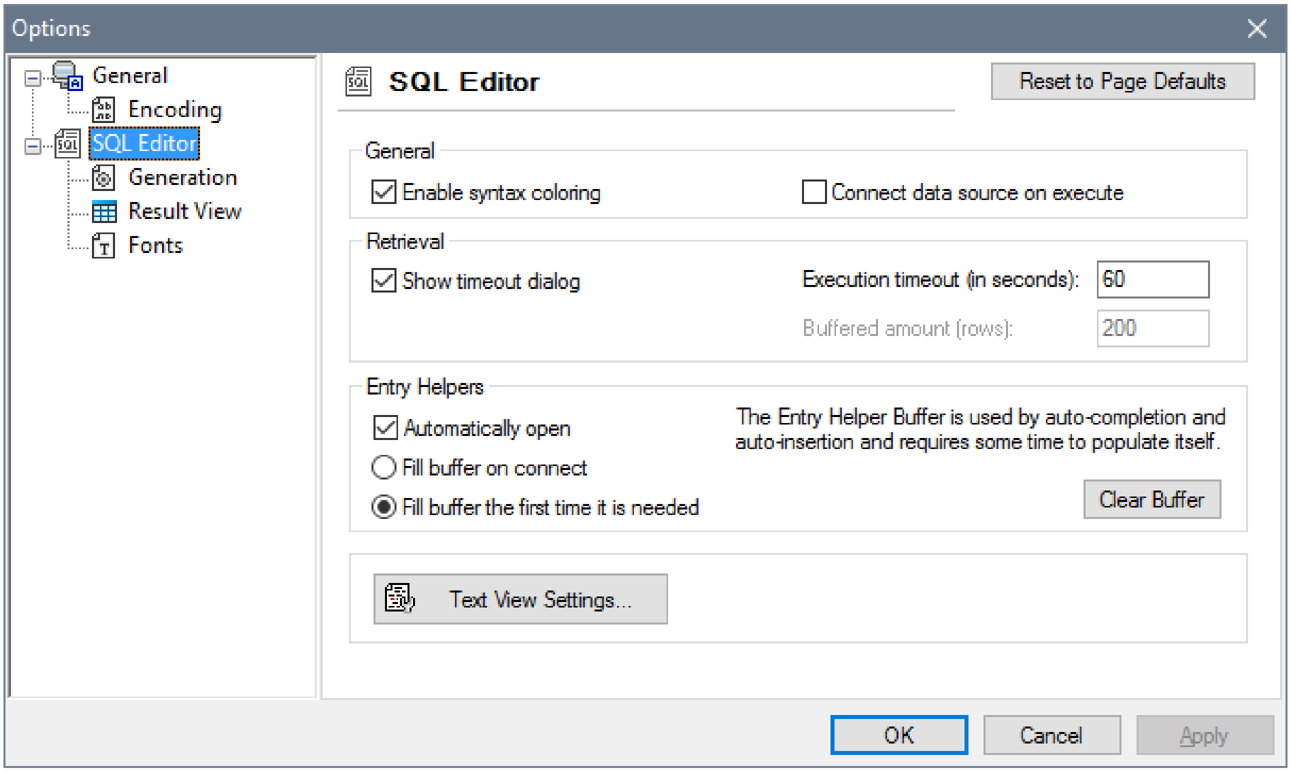
The following icons in the toolbar of the Database Query window are used when working with the SQL Editor:

| Icon | Command | Does this |
|---|---|---|

| | | |
|---|---|---|
| | **Execute Query** | Executes currently selected SQL statement. If script contains multiple statements and none is selected, then all are executed. |
| | **Import SQL File** | Opens an SQL file in the SQL Editor. |
| | **Export SQL File** | Saves SQL queries to an SQL file. |
| | **Undo** | Undoes an unlimited number of edits in SQL Editor. |
| | **Redo** | Redoes an unlimited number of edits in SQL Editor. |
| | **Options** | Open the Options dialog of SQL Editor. |
| | **Open Query in DatabaseSpy** | Opens the SQL script in Altova's DatabaseSpy application. |

## SQL Editor options

Clicking the **Options** icon in the Database Query toolbar pops up the Options dialog (*screenshot below*). Select a category of settings in the left-hand pane, and then the the options on that page can be selected. Click the **Reset to Page Defaults** button to reset the options on that page to their original settings.



General
☑ Enable syntax coloring
□ Connect data source on execute

Retrieval
☑ Show timeout dialog
Execution timeout (in seconds): 60
Buffered amount (rows): 200

Entry Helpers
☑ Automatically open
○ Fill buffer on connect
● Fill buffer the first time it is needed

The Entry Helper Buffer is used by auto-completion and auto-insertion and requires some time to populate itself.

*General > Encoding:*

Options for setting the encoding of new SQL files, of existing SQL files for which the encoding cannot be detected, and for setting the Byte Order Mark (BOM). (If the encoding of existing SQL files can be detected, the files are opened and saved without changing the encoding.)

*SQL Editor*
Options for toggling syntax coloring and data source connections on execution on/off. A timeout can be set for query execution, and a dialog to change the timeout can also be shown if the specified time is exceeded. Entry helpers refer to the entry helpers that appear as part of the auto-completion feature. When you type an SQL statement, the editor displays a list of context-sensitive auto-completion suggestions. These suggestions can be set to appear automatically. If the automatic display is switched off, then you can ask for an auto-completion suggestion in SQL Editor by pressing **Ctrl+Spacebar**. The buffer for the entry helper information can be filled either on connection to the data source or the first time it is needed. The Text View settings button opens a dialog in which you can specify settings such as indentation and tab size of text in the SQL Editor.

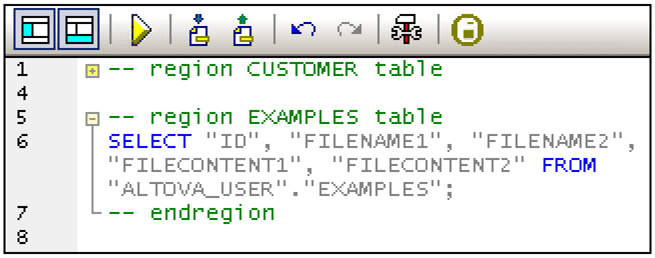
*SQL Editor > SQL Generation*
The application generates SQL statements when you drag objects from the Browser pane into the Query pane. Options for SQL statement generation can be set in the SQL generation tab. Use the Database list box to select a database kind and set the statement generation options individually for the different database kinds you are working with. Activating the *Apply to all databases* check box sets the options that are currently selected for all databases. Options include appending semi-colons to statements and surrounding identifiers with escape characters.

*SQL Editor > Result View*
Options to configure the Result tab.

*SQL Editor > Fonts*
Options for setting the font style of the text in the Text Editor and in the Result View.

## Definition of regions in an SQL script

Regions are sections in SQL scripts that are marked and declared to be a unit. Regions can be collapsed and expanded to hide or display parts of the script. It is also possible to nest regions within other regions. Regions are delimited by `--region` and `--endregion` comments, respectively, before and after the region. Regions can optionally be given a name, which is entered after the `--region` delimiter (*see screenshot below*).



```
1   ⊞ -- region CUSTOMER table
4
5   ⊟ -- region EXAMPLES table
6     SELECT "ID", "FILENAME1", "FILENAME2",
      "FILECONTENT1", "FILECONTENT2" FROM
      "ALTOVA_USER"."EXAMPLES";
7     -- endregion
8
```

To insert a region, select the statement/s to be made into a region, right-click, and select **Insert Region**. The expandable/collapsible region is created. Add a name if you wish. In the screenshot above, also notice the line-numbering. To remove a region, delete the two `--region` and `--endregion` delimiters.

## Insertion of comments in an SQL script

Text in an SQL script can be commented out. These portions of the script are skipped when the script is executed.

- To comment out a block, mark the block, right-click, and select **Insert/Remove Block Comment**. To remove the block comment, mark the comment, right-click and select **Insert/Remove Block Comment**.

- To comment out a line or part of a line, place the cursor at the point where the line comment should start, right-click, and select **Insert/Remove Line Comment**. To remove the line comment, mark the comment, right-click and select **Insert/Remove Line Comment**.

### Use of bookmarks

Bookmarks can be inserted at specific lines, and you can then navigate through the bookmarks in the document. To insert a bookmark, place the cursor in the line to be bookmarked, right-click, and select **Insert/Remove Bookmark**. To go to the next or previous bookmark, right-click, and select **Go to Next Bookmark** or **Go to Previous Bookmark**, respectively. To remove a bookmark, place the cursor in the line for which the bookmark is to be removed, right-click, and select **Insert/Remove Bookmark**. To remove all bookmarks, right-click, and select **Remove All Bookmarks**.

## 11.8.5 Query Pane: Working With

After connecting to a database, an SQL script can be entered in the SQL Editor and executed. This section describes:

- How an SQL script is entered in the SQL Editor.
- How the script is executed in the Database Query window.

The following icons are referred to in this section:

| | | |
|---|---|---|
| ▷ | **Execute Query** | Executes currently selected SQL statement. If script contains multiple statements and none is selected, then all are executed. |
| | **Import SQL File** | Opens an SQL file in the SQL Editor. |

### Creating SQL statements and scripts in the SQL Editor

The following GUI methods can be used to create SQL statements or scripts:

- *Drag and drop:* Drag an object from the Browser pane into the SQL Editor. An SQL statement is generated to query the database for that object.
- *Context menu:* Right-click an object in the Browser pane and select **Show in SQL Editor | Select**.
- *Manual entry:* Type SQL statements directly in SQL Editor. The Auto-completion feature can help with editing.
- *Import an SQL script:* Click the Import SQL File icon in the toolbar of the Database Query window.

### Executing SQL statements

If the SQL script in the SQL Editor has more than one SQL statement, select the statement to execute and click the **Execute** icon in the toolbar of the Database Query window. If no statement in the SQL script is selected, then all the statements in the script are executed. The database data is retrieved and displayed as a grid in the Results tab 1059. Messages about the execution are displayed in the Messages tab 1059.

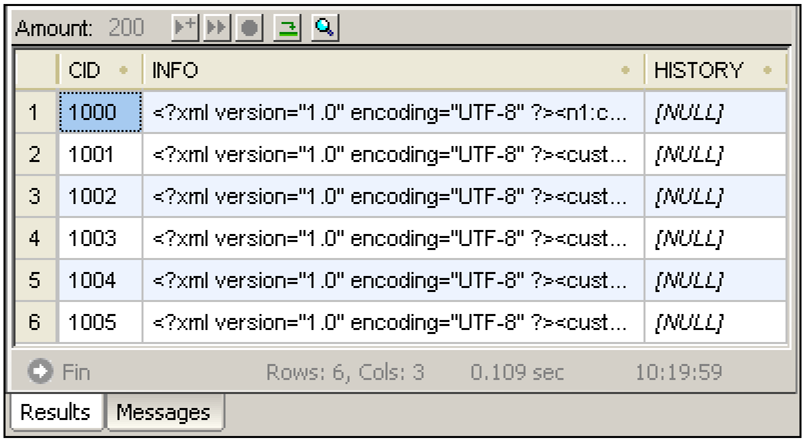
## 11.8.6 Results and Messages Pane

The Results/Messages pane has two tabs:

- The Results tab 1059 shows the data that is retrieved by the query.
- The Messages tab 1059 shows messages about the query execution.

### Results tab

The data retrieved by the query is displayed in the form of a grid in the Results tab (*screenshot below*).



Amount: 200

| | CID | INFO | HISTORY |
|---|---|---|---|
| 1 | 1000 | <?xml version="1.0" encoding="UTF-8" ?><n1:c... | [NULL] |
| 2 | 1001 | <?xml version="1.0" encoding="UTF-8" ?><cust... | [NULL] |
| 3 | 1002 | <?xml version="1.0" encoding="UTF-8" ?><cust... | [NULL] |
| 4 | 1003 | <?xml version="1.0" encoding="UTF-8" ?><cust... | [NULL] |
| 5 | 1004 | <?xml version="1.0" encoding="UTF-8" ?><cust... | [NULL] |
| 6 | 1005 | <?xml version="1.0" encoding="UTF-8" ?><cust... | [NULL] |

Fin Rows: 6, Cols: 3 0.109 sec 10:19:59

The following operations can be carried out in the Results tab, via the context menu that pops up when you right-click in the appropriate location in the Results tab:

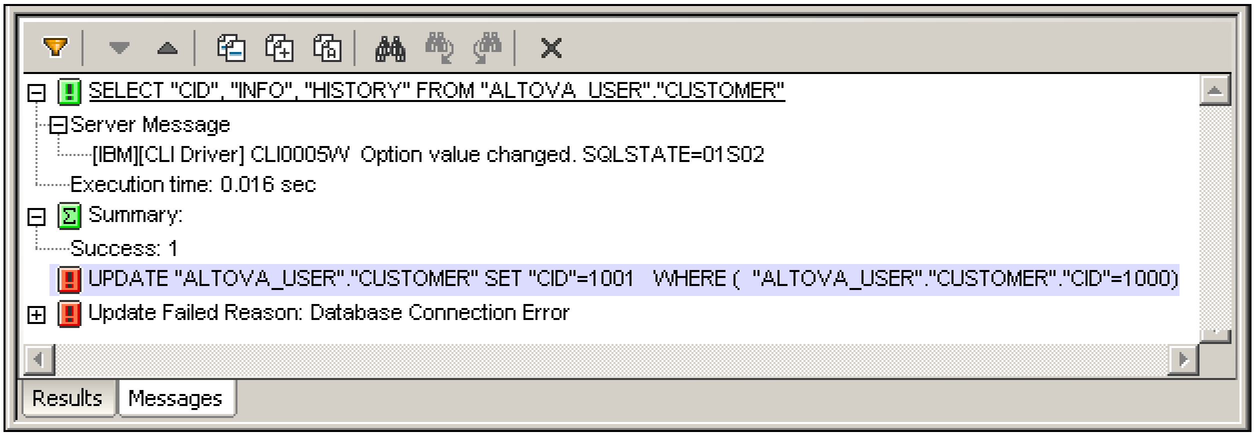
- Sorting on a column: Right-click anywhere in the column on which the records are to be sorted, then select **Sorting | Ascending/Descending/Restore Default**.
- Copying to the clipboard: This consists of two steps: (i) selecting the data range; and (ii) copying the selection. Data can be selected in several ways: (i) by clicking a column header or row number to select the column or row, respectively; (ii) by selecting individual cells (use the **Shift** and/or **Ctrl** keys to select multiple cells); (iii) by right-clicking a cell and selecting **Selection | Row/Column/All**. After making the selection, right-click and select **Copy Selected Cells**. This copies the selection to the clipboard, from where it can be pasted into another application. To copy the header together with the cells, use the command **Copy Selected Cells with Header**.

The Results tab has the following toolbar icons:

| | | |
|---|---|---|
| | **Go to Statement** | Highlights the statement in the SQL Editor that produced the current result. |
| | **Find** | Finds text in the Results pane. XML document content is also searched. |

### Messages tab

The Messages tab provides information on the previously executed SQL statement and reports errors or warning messages.



SELECT "CID", "INFO", "HISTORY" FROM "ALTOVA_USER"."CUSTOMER"
Server Message
[IBM][CLI Driver] CLI0005W Option value changed. SQLSTATE=01S02
Execution time: 0.016 sec
Summary:
Success: 1
UPDATE "ALTOVA_USER"."CUSTOMER" SET "CID"=1001 WHERE ( "ALTOVA_USER"."CUSTOMER"."CID"=1000)
Update Failed Reason: Database Connection Error

The toolbar of the Messages tab contains icons that enable you to customize the view, navigate it, and copy messages to the clipboard. The Filter icon enables the display of particular types of messages to be toggled on or off. The **Next** and **Previous** icons lets you step through the list, downwards and upwards, respectively. Messages can also be copied with or without their child components to the clipboard, enabling them to be pasted in documents. The Find function enables you to specify a search term and then search up or down the listing for this term. Finally, the **Clear** icon clears the contents of the Report pane.

**Note:** These toolbar icon commands are also available as context menu commands.

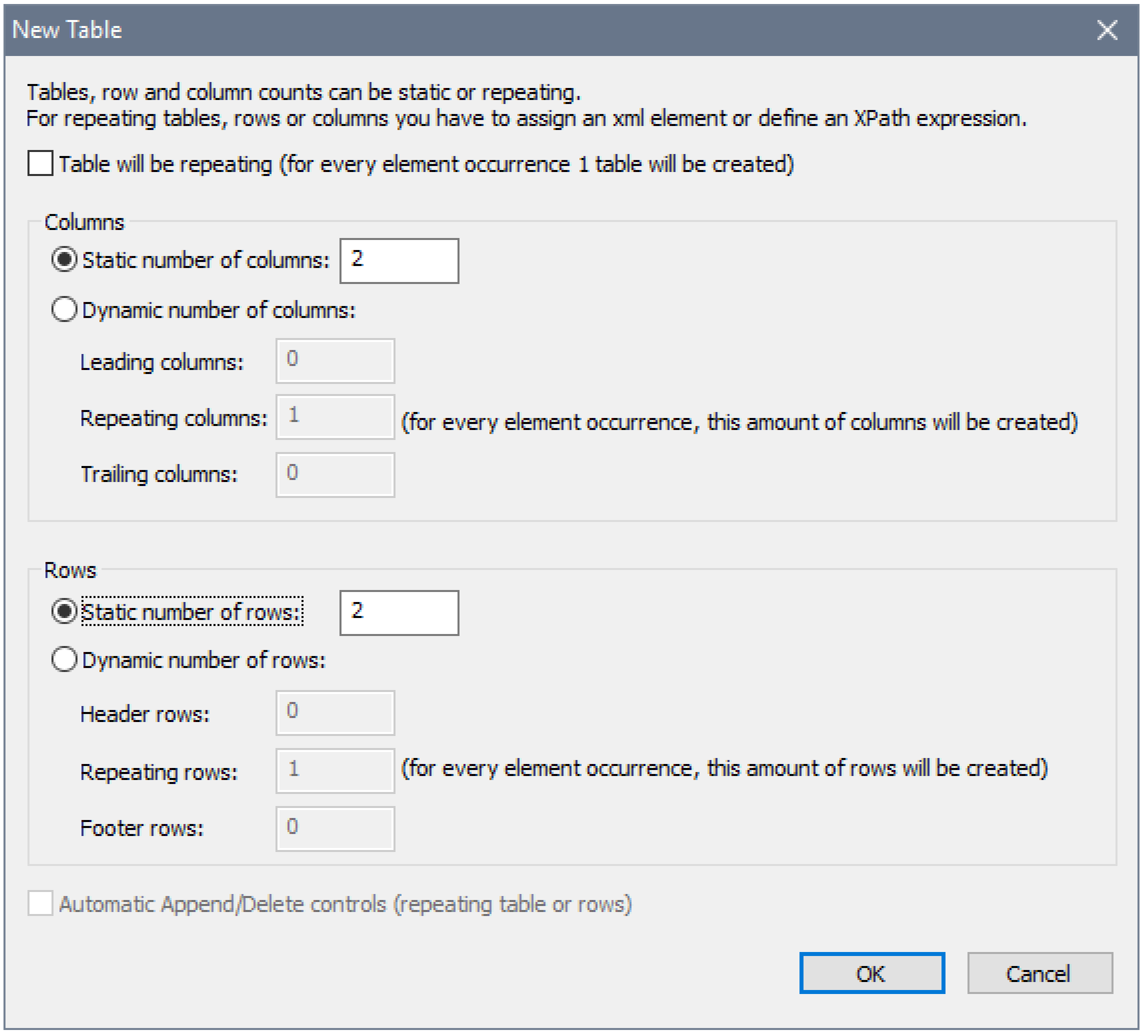
# 12 Design Components

This section describes a range of design objects that can be placed on a page, most of which are controls that can be placed on a page, as well as advanced features that can be defined for a page and/or the project.

- Tables 1062, which explains how to use the different types of table that can be used in a design
- Images 1090, which describes the powerful image handling capabilities of MobileTogether
- Audio, Video 1108, which provides an overview of  the audio-video features of MobileTogether
- NFC 1118, which describes how NFC messages can be sent and received, and set up to be processed further
- Push Notifications 1125, which describes how to set up the sending and receiving of push notifications in solutions
- MQTT 1135, which describes how to set up your MT solution to publish and subscribe to MQTT messages and how to set up actions when an MQTT message is received
- Charts 1171, which explains how charts can be configured in the design
- Control Templates 1200, which describes how to create a template that can be used at multiple locations in the design
- Style Sheets 1318, which describes how global styles can be applied at the project, page, table, and control level
- Rich Text 1225, which describes how the Rich Text control 1225 and Rich Text style sheets 1226 are used to display and edit rich text in a solution.
- Solutions for Authenticated Users 1237, which describes how to set up authentication so users can be directed to a solution on another MobileTogether Server without having to log in a second time
- Hyperlinking to Solutions 1239, which describes mechanisms for starting a solution from (i) another solution, or (ii) a link in an email

# 12.1 Tables

You can insert tables into your design by dropping the Table control[614] at the location where you want the table. For each Table control[614], one table is created. When you drop the control, the New Table dialog (*screenshot below*) appears. Here you can specify the table's type and whether the table should have a header and/or footer.



Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

□ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 2

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 2

○ Dynamic number of rows:

Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

□ Automatic Append/Delete controls (repeating table or rows)

**Note:** You can nest tables within other tables by dropping them in the desired location. You can then use the structure-modification commands in the Table menu[1637] (or the table's context menu[1087]) to correctly set the nested table in its parent table and obtain the desired overall table structure. For example, you can use the **Split** and **Join** commands to get desired colspans and rowspans.

## Types of tables

A table's type is specified at the time when it is created. You can change the type of some tables subsequently.

There are four types of tables:

- Static tables[1063] have a fixed number of rows and columns and are useful for presenting data in neatly ordered rows and columns. In the New Table dialog, specify the number of static rows and columns. You can modify the structure at any later time. Static tables are different than dynamic tables in that they have a fixed number of rows and columns that are independent of the underlying data, whereas in dynamic tables the number of either rows or columns depend on the underlying data (*see below*).
- Repeating tables[1065] are associated with a data-source element. For each instance of this element, one table is created. As a result, the table repeats for each instance of the element. Since each repeating table is created as a row, repeating tables expand vertically. Repeating tables can contain tables with repeating columns (*see fourth point of this list*).
- Dynamic tables with repeating rows[1069] are similar to repeating tables, except that here it is the row—not the table—that repeats with every instance of the element with which the row is associated. Tables with repeating rows can contain tables with repeating columns.
- Dynamic tables with repeating columns[1074] are of two kinds: (i) where the rows are static, and (ii) where the rows are dynamic (repeating). Note that, if both the column and row repeat, each must be associated with a different repeating element from the page source.

### Table structure and properties

The internal structure of the table can be modified at any subsequent time by using the structure-modification commands in the Table menu[1637] (or the table's context menu[1087]). Properties such as background colors, padding, text color for the entire table and for individual columns, rows, and cells are specified by selecting the respective table component and assigning its properties in the Styles & Properties Pane[274].

For more information, see the sections: Table Properties[1078] and Table Context Menu[1087]. Also see the description of the Table control[614].

### In this section

This section is organized as follows:

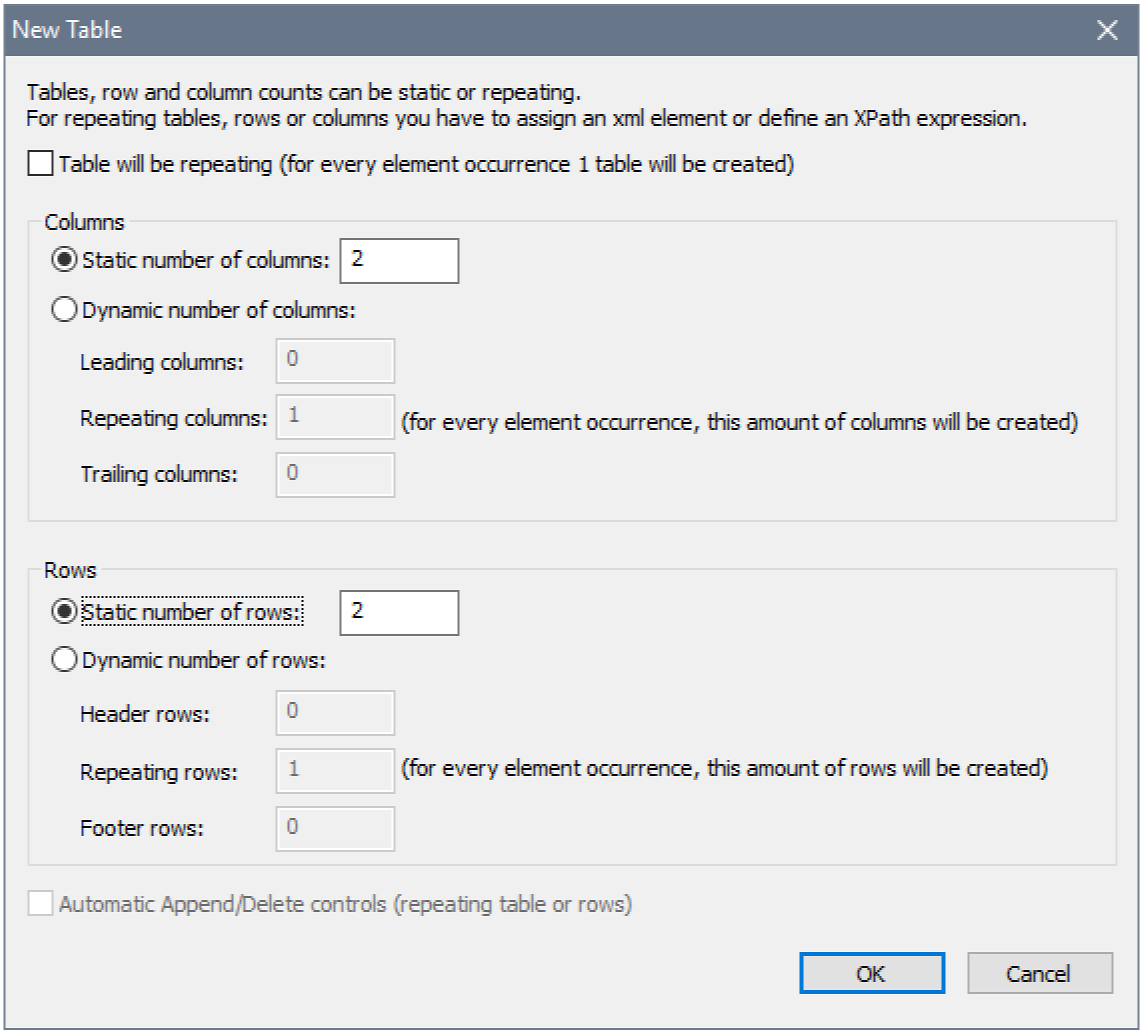
- Static Tables[1063]
- Repeating Tables[1065]
- Dynamic Table with Repeating Rows[1069]
- Dynamic Table with Repeating Columns[1074]
- Table Properties[1078]
- Table Context Menu[1087]

## 12.1.1 Static Tables

Static tables have a fixed number of rows and columns and are structurally independent of the underlying data structure. They are useful for presenting data in neatly ordered rows and columns.

### Table creation and structure

A table is defined as a static table at the time when the Table control[614] is dropped into the design. In the New Table dialog that appears when the control is dropped in the design (*screenshot below*), specify a static number of columns and a static number of rows. On clicking **OK**, a static table with the specified number of columns and rows is inserted. You can now add content to the cells of the table and specify style properties for the table and for individual columns, rows, and cells.



Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

☐ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 2

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 2

○ Dynamic number of rows:

Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

☐ Automatic Append/Delete controls (repeating table or rows)

Table restructuring commands are available in the context menu of the table[1087] and the Table menu[1637].

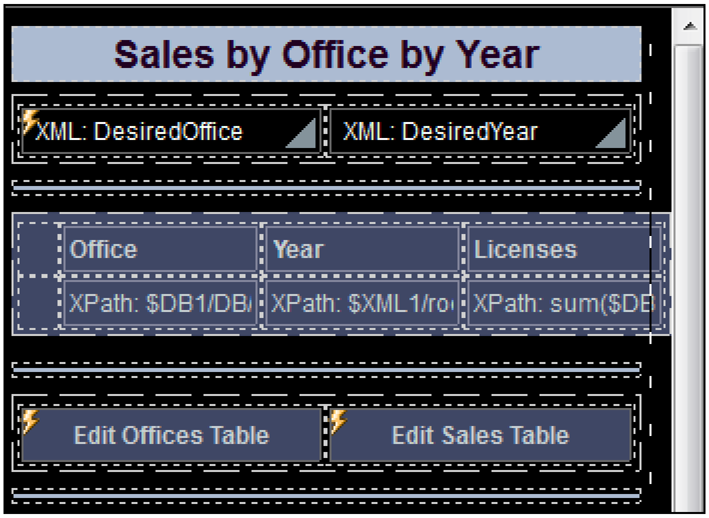
Table formatting properties[1078] are available in the Styles & Properties Pane[274].

## Table contents

The cells of a static table can contain the following:

- Static text
- A node from a page source
- A page control (with or without a link to a page source node)
- A nested table

Cell content is typically a page control[404]. The screenshot below shows three static tables in a design.



All the cells in these tables, except two cells (which are empty), contain one control[404] each.

- The two cells of the first table contain combo boxes[459].
- The second table contains two empty cells (used for spacing) and six cells with one Label control[554] each.
- The third table contains two buttons[289]. (The lightning symbols on some controls indicate that the controls have actions[667] defined for them.)

## 12.1.2 Repeating Tables

Repeating tables work as follows:

- A repeating element in the page source is associated with the repeating table (*see examples below*). For example, the **`//Department/Person`** element is a repeating element if a **`Department`** element has more than one **`Person`** element.
- For each instance of the repeating element one table will be generated, and these tables will be added in a vertical sequence. So, in our example, one table will be generated for each **`Person`** element, with every subsequent table being placed below the preceding table.
- The repeating table can be designed to contain any number of static rows and columns. Columns do not need to be static; they can be repeating.
- The entire table design is repeated for each instance of the repeating element. So, if the table design contains two rows, then a table with two rows will be created for each instance of the repeating element.
- The content of the table will be dynamic. This means that XPath expressions inside each table will have the associated instance of the repeating element as its context node (*see examples below*).
- An Append/Delete control can be added to the design of the repeating table. This control enables end users to add a new instance of the repeating table or to delete any of the individual repeating tables. When a table is added or deleted, the underlying data is correspondingly modified. For example: (i) if a new table is added to our example data structure, then, correspondingly, a new **`Person`** element is added to the data; (ii) if a table is deleted, then the corresponding **`Person`** element is deleted from the data. These modifications can be saved back to the data source, thus enabling end users to modify the data source.

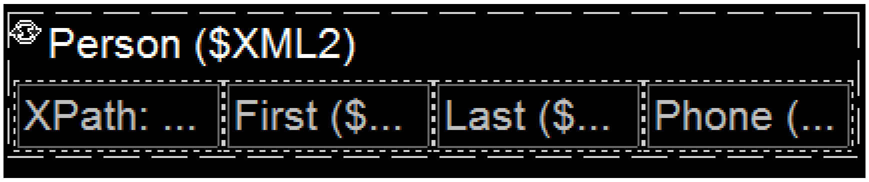
A repeating table is defined at the time when the Table control[614] is dropped into the design. Note, however, that a repeating table can also be created by converting a static table to a repeating table. To do this, first set the static table's `Repeating` property to **`true`**, and then assign the repeating element that you want to associate with the repeating table.

## Example

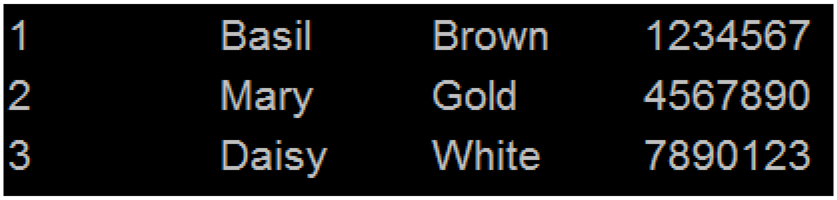
A **`Person`** element in the page source contains, say, **`First`**, **`Last`**, and **`Phone`** child elements. The **`Person`** element can occur multiple times (its instances). We would like to create a table like this:

| | **<First>** | **<Last>** | **<Phone>** |
|---|---|---|---|
| **<Person>** | | | |
| **<Person>** | | | |
| **<Person>** | | | |

Since the **`Person`** element repeats, we can create a repeating table that is associated with the **`Person`** element and specify, in the New Table dialog (*see below*), that the table has one row and four columns. Inside this table the context node is **`Person`**. In each column, we make page source links to the respective child nodes (*see screenshot below*). The first column contains an XPath expression to number the current **`Person`** element, for example: **`count(preceding-sibling::*)+1`**. The design would look something like this:



The generated table would look something like this:



| | | | |
|---|---|---|---|
| 1 | Basil | Brown | 1234567 |
| 2 | Mary | Gold | 4567890 |
| 3 | Daisy | White | 7890123 |

**Note:** A data stream can be generated from an XPath/XQuery expression and can be used as a data source. However, this kind of data source is created for current use only and cannot be accessed as a page source for use elsewhere in the document.

## Difference between a repeating table and a dynamic table with repeating rows

A *repeating table*[1065] is different than a *dynamic table with repeating rows*[1069] in the following ways:

- In a *repeating table*[1065], it is the **entire table** that is associated with the repeating element of the data source. A new table is generated for each instance of the repeating data element.
- In a *dynamic table with repeating rows*[1069], a **table row group** within the table is associated with the repeating element of the data source. One table row group is generated for each instance of the repeating element.

This difference has two design effects:

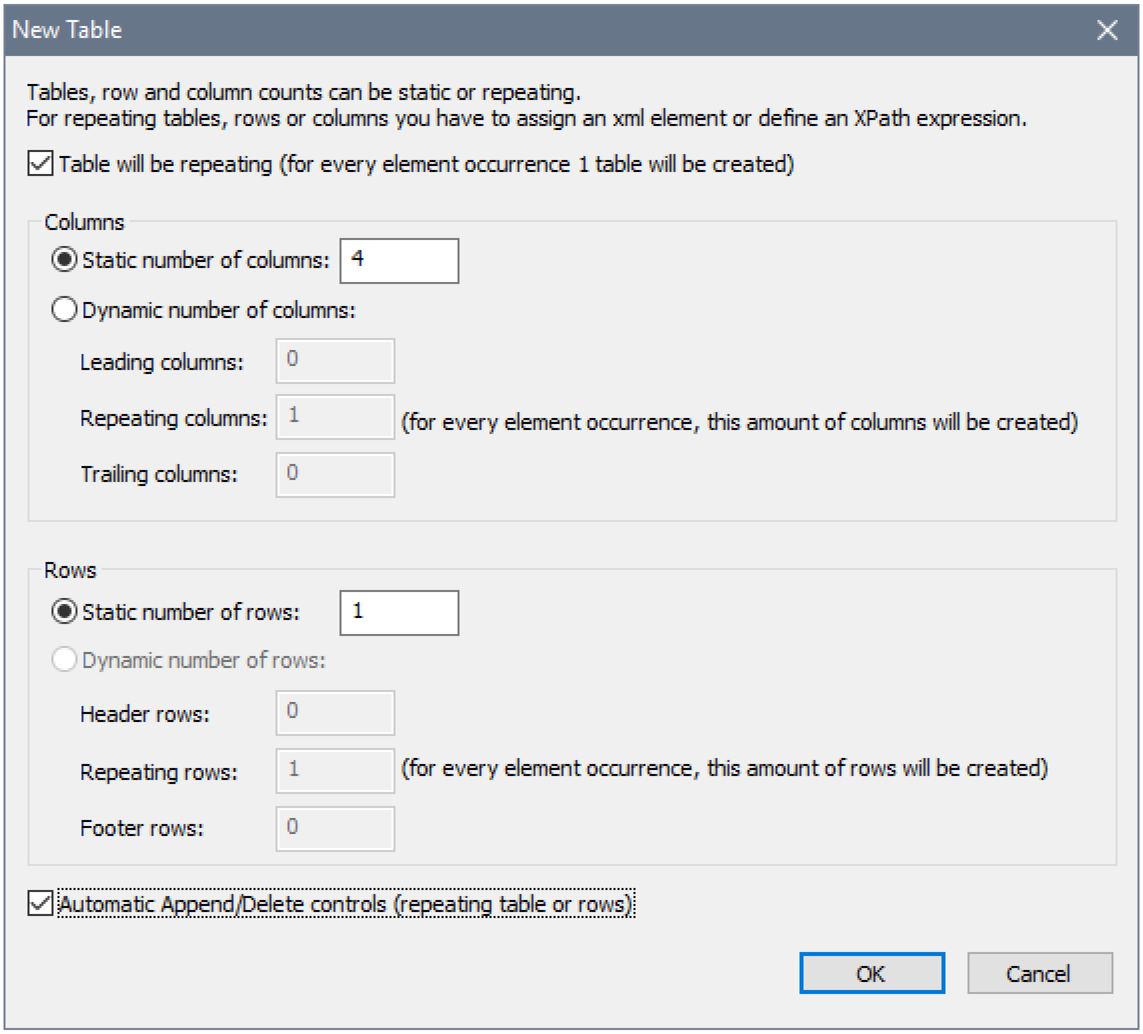
- A *dynamic able with repeating rows*[1069] can have a header and/or footer that applies to the entire table. However, if a header/footer is required for a *repeating table*[1065], it should be added outside the repeating table. If added inside the table, then the header/footer too will repeat with each repeat of the table.
- Since tables are typically rendered with space above and below them in device displays, there might be vertical space between a pair of *repeating tables*[1065].

To change a static table to a repeating table after the table has been created, change its Repeating property[1078] to `true`, then associate the table with a repeating element from the page source.

## Creating a repeating table

Set up a repeating table as follows:

1. In the New Table dialog that appears when the Table control is dropped in the design (*screenshot below*), check *Table will be repeating* to make the table repeating. Note that it is the table that repeats for every instance of a repeating data row.



New Table

Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

☑ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 4

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

● Static number of rows: 1

○ Dynamic number of rows:

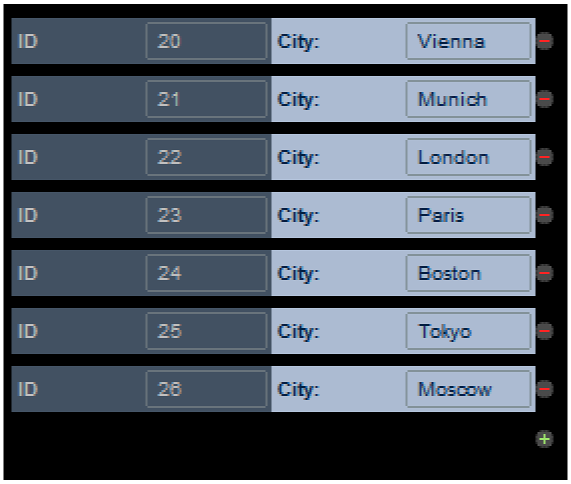
Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

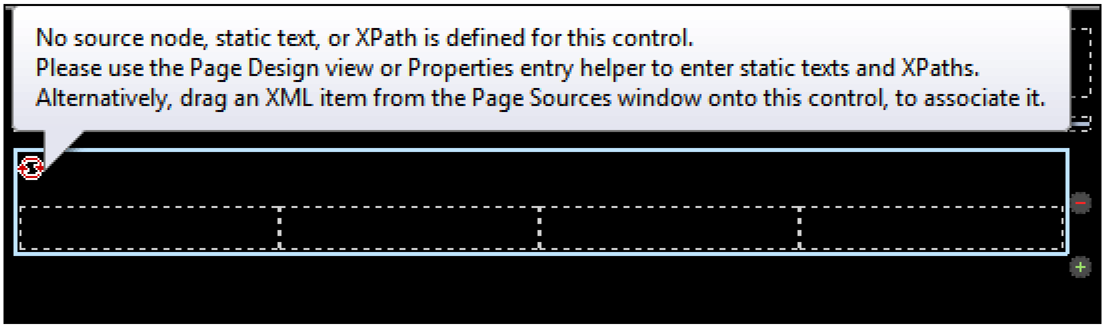
☑ Automatic Append/Delete controls (repeating table or rows)

2. Specify the number of static columns and rows that the repeating table will have. You can subsequently add columns and rows to the repeating table via the table's context menu[1087].
3. Specify whether automatic Append/Delete controls are to be added. If added, each repeating table in the solution, effectively a row, will have a **Delete** button, and the entire repeating table structure will have an **Append** button to append a repeating table (to the structure (*see the screenshot of a simulated solution below*).



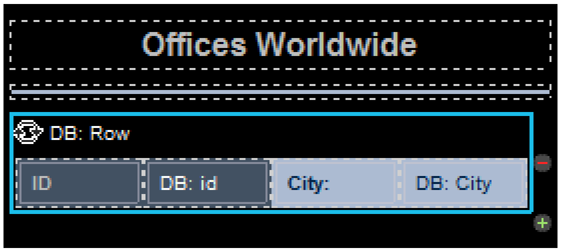
| ID | 20 | City: | Vienna |
|---|---|---|---|
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

4. On clicking **OK** in the New Table dialog, the table is added to the design. The repeating table must now be associated with the repeating element from the page source (*see screenshot below*).



No source node, static text, or XPath is defined for this control.
Please use the Page Design view or Properties entry helper to enter static texts and XPaths.
Alternatively, drag an XML item from the Page Sources window onto this control, to associate it.

5. Associate a repeating element with the repeating table by dragging and dropping the element from the Page Sources Pane[270] into the table.
6. You can now add content to the cells of the table. The context node for XPath expressions within table cells is the element node that is associated with the repeating table (*see previous step*). To use the context node, XPath expressions in table cells should be relative to the context node. Cell content can be a nested table (static or dynamic), or a page control (with or without a link to a page source node), or even page source nodes. When a page source node is dropped into a cell, the data in that cell will be editable. In the screenshot below, four controls have been added (from left to right): a label, edit field, label, and edit field.

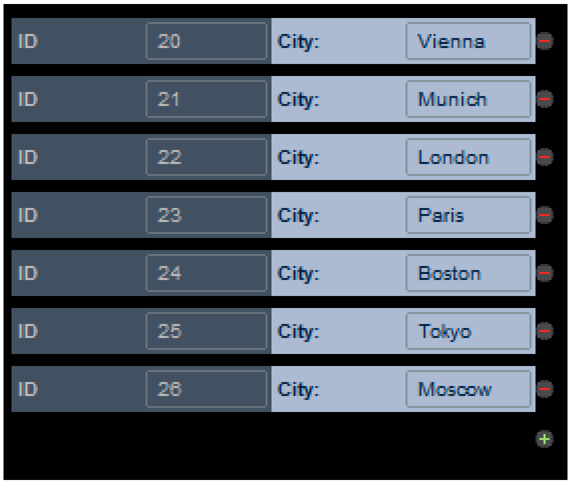


Offices Worldwide

DB: Row

| ID | DB: id | City: | DB: City |
|---|---|---|---|

This repeating table produces the following repeating structure in the MobileTogether solution.



| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

Table restructuring commands are available in the context menu of the table[1087]. Table formatting properties[1078] are available in the Styles & Properties Pane[274]. See also Repeating Rows[1069] and Repeating Columns[1074].

## 12.1.3 Dynamic Table with Repeating Rows

A dynamic table with repeating rows works as follows:

- A repeating row is associated with a repeating element of a page source.
- When the table is rendered, the number of rows in it will dynamically match the number of instances of the repeating element. Each table row will correspond to one instance of the repeating element.
- When you define a table with repeating rows, you can specify how many rows will be repeated for each element occurrence. For example, you could specify that each **repeating row group** contains two rows. In this case, two rows are generated for each instance of the repeating element. Since the table unit that corresponds to the repeating data element can contain more than one row, we refer to this unit as a **table row group**. So, more precisely, it is the row group that repeats.
- The context node of each repeating row group will be the specific instance of the repeating element in the data source
- An Append/Delete control can be added to the table. This enables end users to add a new row group and to delete individual row groups. Each addition/deletion will correspond to an instance of the repeating data element. The end user can save modifications back to the data source.
- You can mix static and repeating rows. Consequently, a static row can be placed between two repeating row groups.

- For each table with repeating rows, you can add a header or footer via context menu commands. In this case, the entire table would have a header or footer, and the repeating row groups would be generated within the body of the table.
- If a table with repeating rows has a header or footer, you can convert the entire group of components into a repeating table via the context menu. In this case, all the components of the repeating table will be repeated for each instance of the repeating data element.
- A dynamic table can contain more than one table row group.

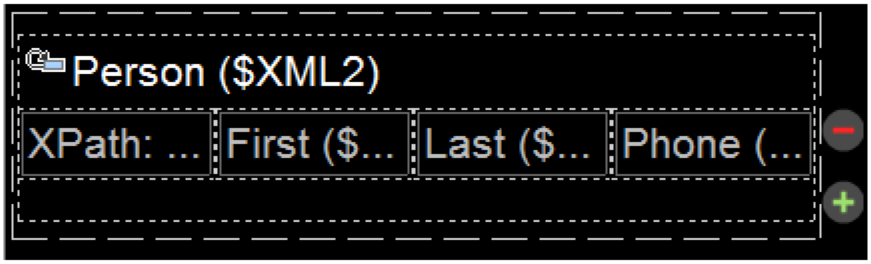
A dynamic table with repeating rows is defined at the time when the Table control[614] is dropped into the design.

## Example

A **Person** element in the page source contains a repeating structure (say, of **First**, **Last**, and **Phone** elements). The **Person** element can occur multiple times (its occurrences). If a table row group is associated with the **Person** element, then the table will be generated with as many table row groups as there are **Person** elements (*see table below*). If the number of **Person** elements in the page source changes, then the number of rows in the table changes automatically.

| | <First> | <Last> | <Phone> |
|---|---|---|---|
| <Person> | | | |
| <Person> | | | |
| <Person> | | | |

The design shown in the screenshot below contains a table that contains a single repeating table row group. The row group is associated with the **Person** element, and it consists of one row and four columns.



For each occurrence of the element associated with the row group, the entire row group is generated. XPath expressions inside the row group are resolved with the current occurrence of the associated element as the context node.

**Note:** A data stream can be generated from an XPath/XQuery expression and can be used as a data source. However, this kind of data source is created for current use only and cannot be accessed as a page source for use elsewhere in the document.

## Difference between a repeating table and a dynamic table with repeating rows

A *repeating table*[1065] is different than a *dynamic table with repeating rows*[1069] in the following ways:

- In a *repeating table*[1065], it is the **entire table** that is associated with the repeating element of the data source. A new table is generated for each instance of the repeating data element.
- In a *dynamic table with repeating rows*[1069], a **table row group** within the table is associated with the

repeating element of the data source. One table row group is generated for each instance of the repeating element.

This difference has two design effects:

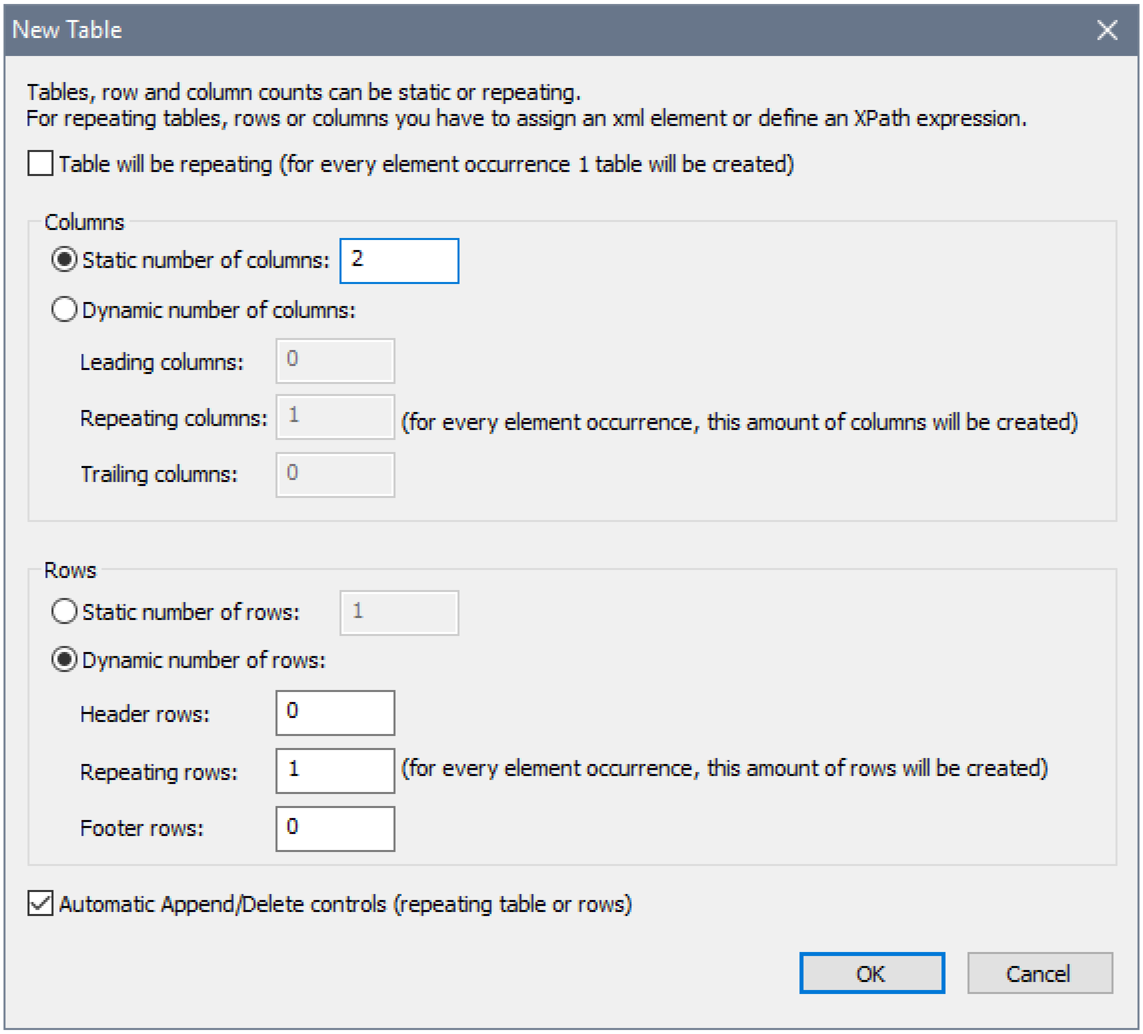
- A *dynamic able with repeating rows*[1069] can have a header and/or footer that applies to the entire table. However, if a header/footer is required for a *repeating table*[1065], it should be added outside the repeating table. If added inside the table, then the header/footer too will repeat with each repeat of the table.
- Since tables are typically rendered with space above and below them in device displays, there might be vertical space between a pair of *repeating tables*[1065].

To convert a *repeating table*[1065] to a *table with dynamic rows*[1069], right-click the table row that you wish to convert to a dynamic row, then select **Dynamic or Repeating Table | Convert this Row to Repeating Row**. You can add a header or footer to a repeating row via the context menu command **Dynamic or Repeating Table**.

## Creating a table with repeating rows

Set up a table with repeating rows as follows:

1. In the New Table dialog that appears when the control is dropped (*screenshot below*), make sure that *Table will be repeating* is unchecked. Then select *Dynamic number of rows*. This will create a table that contains one repeating table row group.



New Table

Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

□ Table will be repeating (for every element occurrence 1 table will be created)

Columns

● Static number of columns: 2

○ Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

○ Static number of rows: 1

● Dynamic number of rows:

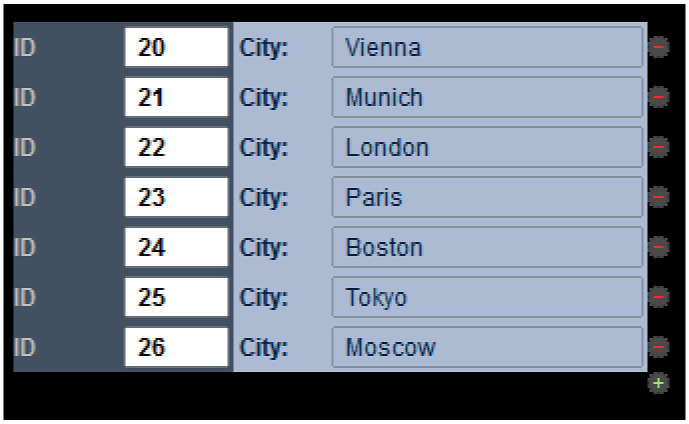
Header rows: 0

Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

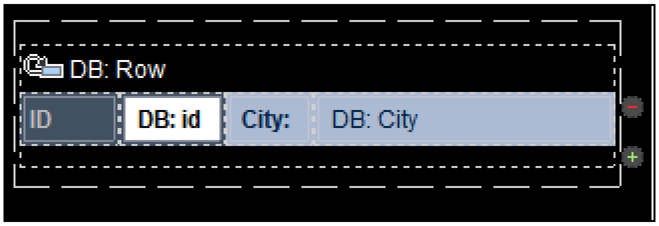
☑ Automatic Append/Delete controls (repeating table or rows)

2. Specify the number of columns that the table will have and the number of rows that the row group will have (these rows together constitute the row group). You can also specify that the table should have a header and/or footer.
3. Specify whether automatic Append/Delete controls are to be added. If added, each repeating row in the solution will have a **Delete** button and the table will have an **Append** button to append a row group (*see the screenshot of a simulated solution below*).



| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

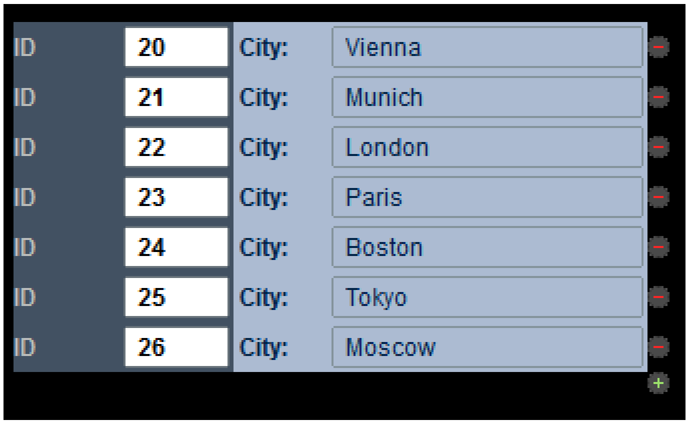
4. On clicking **OK** in the New Table dialog, the table is added to the design.
5. The row group must now be associated with the repeating element from the page source. Associate the row group with a repeating element by dragging and dropping the element from the Page Sources Pane 270 into the table. Each instance of this element will generate a row group in the table.
6. You can now add content to the cells of the table. The context node for XPath expressions within table cells is the specific instance of the element node that is associated with the dynamic row group (*see previous step*). Cell content can be a nested table (static or dynamic), or a page control (with or without a link to a page source node), or even page source nodes. When a page source node is dropped into a cell, the data in that cell will be editable. In the screenshot below, four controls have been added (from left to right): a label, edit field, label, and edit field.



DB: Row

| | | | |
|---|---|---|---|
| ID | DB: id | City: | DB: City |

This dynamic row produces the following structure in the MobileTogether solution.



| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

For information about spanning dynamic columns,see the section Row/Column joining and spanning 1083.

Table restructuring commands are available in the context menu of the table[1087]. Table formatting properties[1078] are available in the Styles & Properties Pane[274]. See also Repeating Tables[1065] and Dynamic Columns[1074].

### Adding additional table row groups

You can use any of the available table-construction mechanisms to add additional table row groups. For example, you can add a static row and convert this to a table row group, or you can add a table with repeating rows.

## 12.1.4 Dynamic Table with Repeating Columns

Within a table row, if a column repeats, then these repetitions can be displayed via the dynamic columns feature. The column is associated with a repeating element of a page source. When the table is rendered, the number of columns will correspond dynamically to the number of instances of the associated element.

Dynamic columns can occur in two types of row contexts:

- *Static rows, dynamic columns:* In this situation, the table grows horizontally rather than vertically (*compare table below with dynamic table with repeating rows*[1069]). In this case, the leading column acts as a "header"; it can contain the names of the rows. (A leading column can be added when the table is first created, but since a leading column is static it can also be created subsequently).

| | **<Person>** | **<Person>** | **<Person>** |
|---|---|---|---|
| **<First>** | | | |
| **<Last>** | | | |
| **<Phone>** | | | |

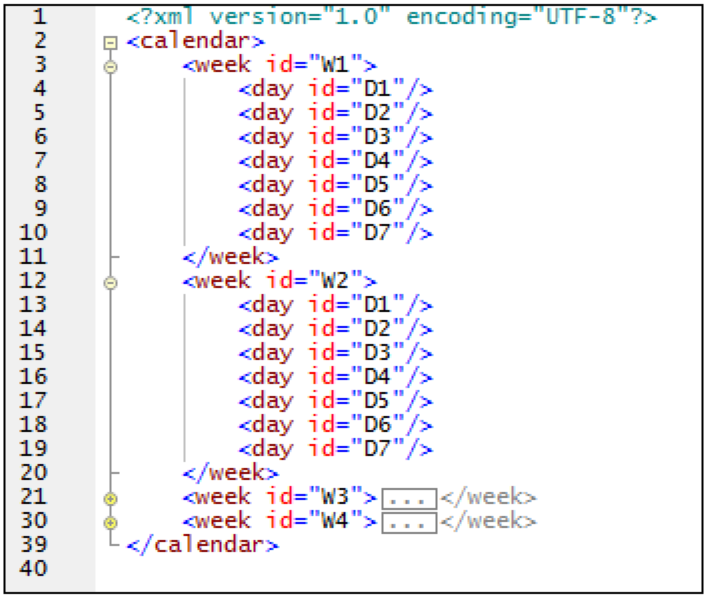
- *Dynamic rows, dynamic columns:* The table can grow both vertically (additional rows) and horizontally (additional columns). To create this kind of table, the number of column element instances in the page source must be the same for all rows. For example, in the table below, each of the four **week** elements (each corresponding to a row in the table) contains exactly three **day** elements (the columns of the table). If any **week** element contains some other number of **day** elements than three, then the table cannot be correctly drawn. Note also that, typically: (i) column elements occur within row elements, both in the page source and in the table design; (ii) the names of row elements are the same and the names of column elements are the same. However, neither of these two points is a necessary condition for building this kind of dynamic table: (i) column elements can occur outside row elements, and (ii) the names of rows/columns can be different.

| | **<day>** | **<day>** | **<day>** |
|---|---|---|---|
| **<week>** | | | |
| **<week>** | | | |
| **<week>** | | | |
| **<week>** | | | |

**Note:** A data stream can be generated from an XPath/XQuery expression and can be used as a data source. However, this kind of data source is created for current use only and cannot be accessed as a page source for use elsewhere in the document.

## Example: dynamic columns for days inside dynamic rows for weeks

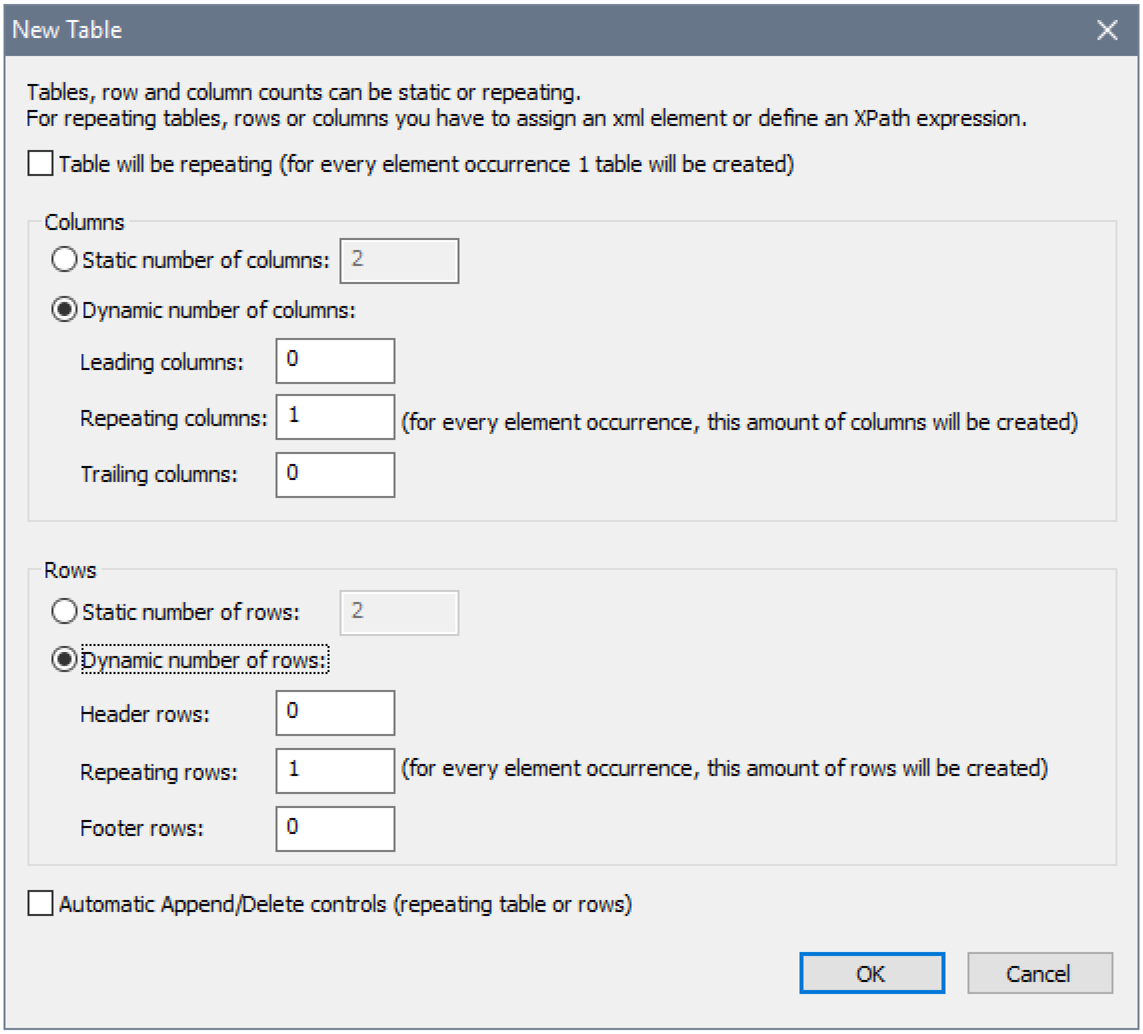
The screenshot below shows a **calendar** element that contains four **week** elements, with each **week** element containing seven **day** elements. We can make a table containing dynamic rows for **week** elements and dynamic columns (inside each **week** element) for the **day** elements. Notice that in the data structure (i) the **day** elements are inside the **week** elements, and (ii) the number of **day** elements inside every **week** element is the same: seven.



```
<?xml version="1.0" encoding="UTF-8"?>
<calendar>
    <week id="W1">
        <day id="D1"/>
        <day id="D2"/>
        <day id="D3"/>
        <day id="D4"/>
        <day id="D5"/>
        <day id="D6"/>
        <day id="D7"/>
    </week>
    <week id="W2">
        <day id="D1"/>
        <day id="D2"/>
        <day id="D3"/>
        <day id="D4"/>
        <day id="D5"/>
        <day id="D6"/>
        <day id="D7"/>
    </week>
    <week id="W3">...</week>
    <week id="W4">...</week>
</calendar>
```

## Creating dynamic columns inside dynamic rows

Drag a table control into the design and drop it at the location where you want to create it. In the New Table dialog (*screenshot below*), select the options for dynamic rows and dynamic columns. Select the number of rows and columns that should be repeated for each instance of the element that, respectively, corresponds to the row and column. Leading/Trailing columns correspond to Header/Footer rows. Note that you can add Append/Delete controls for dynamic rows, but not for dynamic columns.



Tables, row and column counts can be static or repeating.
For repeating tables, rows or columns you have to assign an xml element or define an XPath expression.

□ Table will be repeating (for every element occurrence 1 table will be created)

Columns

○ Static number of columns: 2

● Dynamic number of columns:

Leading columns: 0

Repeating columns: 1 (for every element occurrence, this amount of columns will be created)

Trailing columns: 0

Rows

○ Static number of rows: 2

● Dynamic number of rows:

Header rows: 0

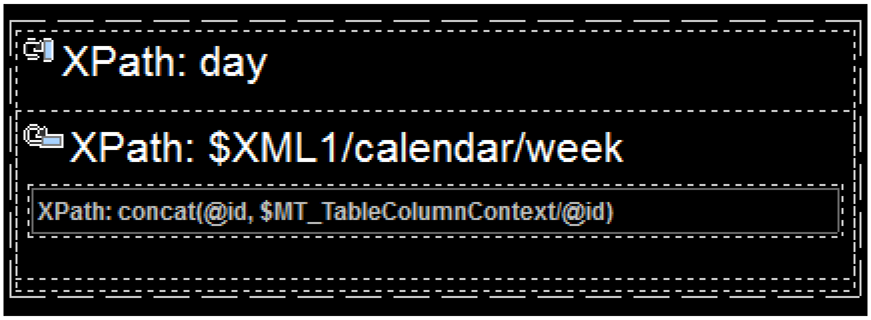
Repeating rows: 1 (for every element occurrence, this amount of rows will be created)

Footer rows: 0

□ Automatic Append/Delete controls (repeating table or rows)

The table will be created in the design. The screenshot below shows the completed design of a table with dynamic rows and columns. Note the following points:

- The fields representing the table's dynamic rows and columns are indicated by icons that, respectively, show a row and column.
- These fields must be associated with the page source nodes that will provide the data for the table's rows and columns. The element associated with the column must, in the page source, be contained within the element associated with the row. In the design, however, notice that the row field is placed within the column field.



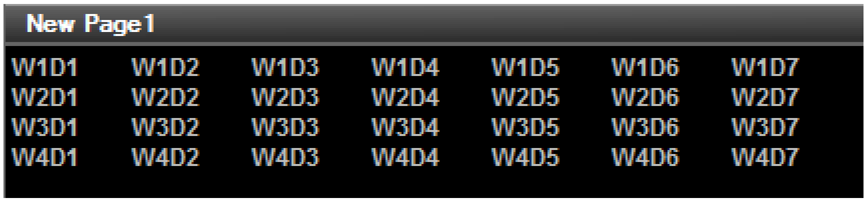
XPath: day

XPath: $XML1/calendar/week

XPath: concat(@id, $MT_TableColumnContext/@id)

- Also note the XPath expressions that are used to associate the row and column fields with page source nodes. The expression that selects the element for the row field must select all instances of the corresponding element. The XPath expression in the screenshot above selects all the **week** child elements of the **calendar** element: **$XML1/calendar/week**. The XPath expression for the column field, however, must select only the column element within the current row. So, an XPath expression such as **$XML1/calendar/week/day** would not work because it would select all the **day** child elements of all **week** elements. Note also that the XPath context node for the column field is the element that is associated with the row. In our example, the context node of the column field is **week**. As a result, an XPath expression of **day** would select the **day** child elements of the current **week** element.
- In the design there is a single cell located at the intersection of the row and column fields. The context node of this cell is the element corresponding to the row field (in our example, the **week** element). XPath expressions in this cell must be constructed within this context. When the table is created, then within each "row element" (the **week** element in our example), a cell is created for each column. The XPath expression is evaluated for each column's cell within the context of the current row. To reach the contents of the element that corresponds to the column field, a special variable is available: `$MT_TableColumnContext`, which, at runtime, contains the element that corresponds to the current column (in our example the current **day** element within the current **week** element). All this is best explained with an example. In the screenshot above, the cell in the design that is at the intersection of the row and column fields contains a label control 554. This control has text that is provided by an XPath expression: **concat(@id,** `$MT_TableColumnContext/@id`**)**. The **concat()** function concatenates two strings: the ID of the current week—obtained by **@id**— and the ID of the current day within the current week—obtained by `$MT_TableColumnContext/@id`. Since the context node of the entire XPath expression is the **week** element (associated with the rows) **@id** provides the value of the current **week/@id** attribute, while the `$MT_TableColumnContext/@id` expression fetches the content of the current `day/@id` attribute within the current **week** element.

The output of the table design that is shown in the screenshot above will look something like this:



New Page1

| | | | | | | |
|---|---|---|---|---|---|---|
| W1D1 | W1D2 | W1D3 | W1D4 | W1D5 | W1D6 | W1D7 |
| W2D1 | W2D2 | W2D3 | W2D4 | W2D5 | W2D6 | W2D7 |
| W3D1 | W3D2 | W3D3 | W3D4 | W3D5 | W3D6 | W3D7 |
| W4D1 | W4D2 | W4D3 | W4D4 | W4D5 | W4D6 | W4D7 |

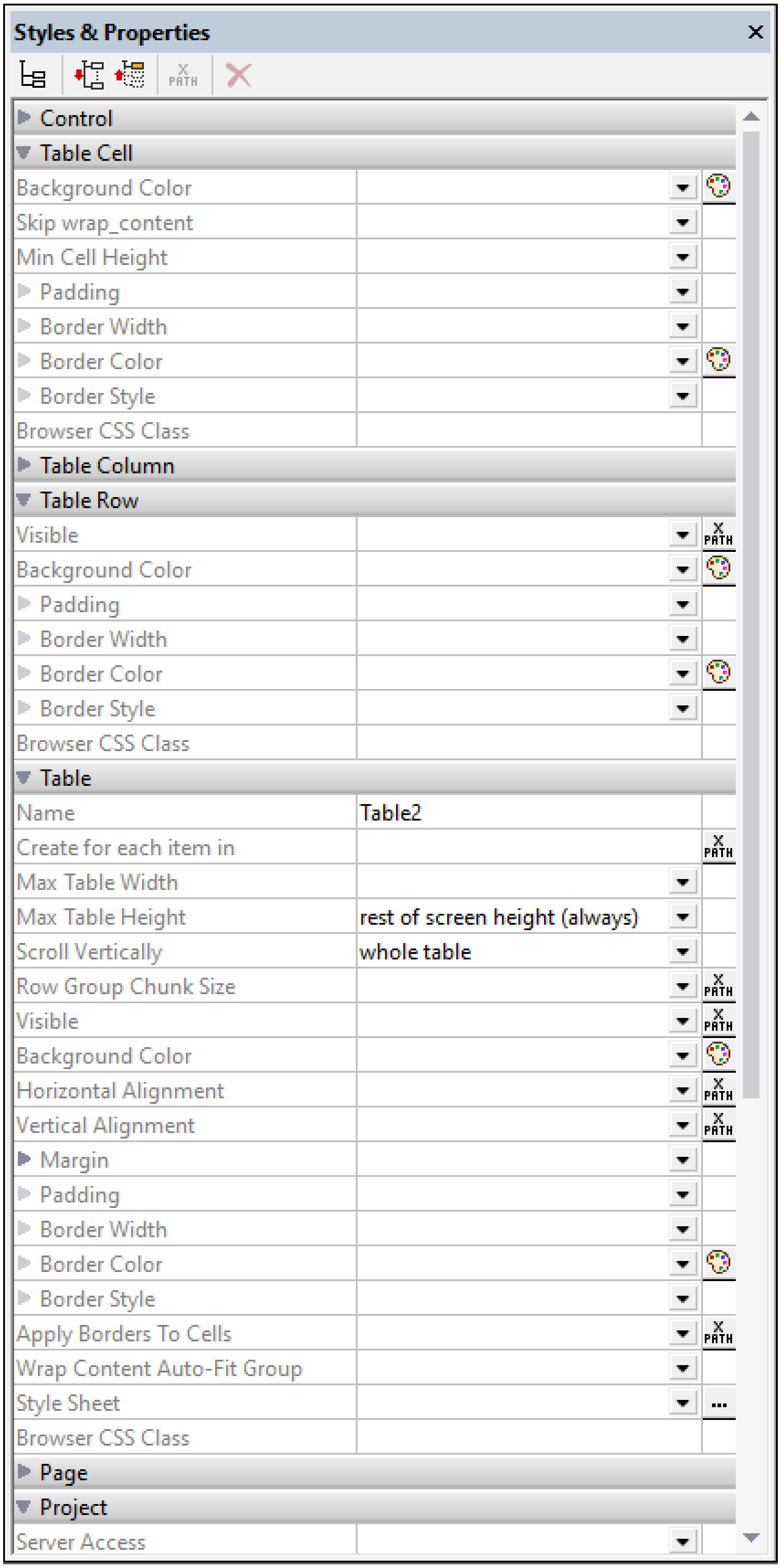
- Each **week** element in the page source is displayed in a row (**W1** to **W4**).
- Each **day** element within a week is displayed within the appropriate column (**D1** to **D7**).
- The concatenation of the two IDs is executed separately for the 28 cells from **W1D1** to **W4D7**.

For information about spanning dynamic columns,see the section Row/Column joining and spanning 1083.

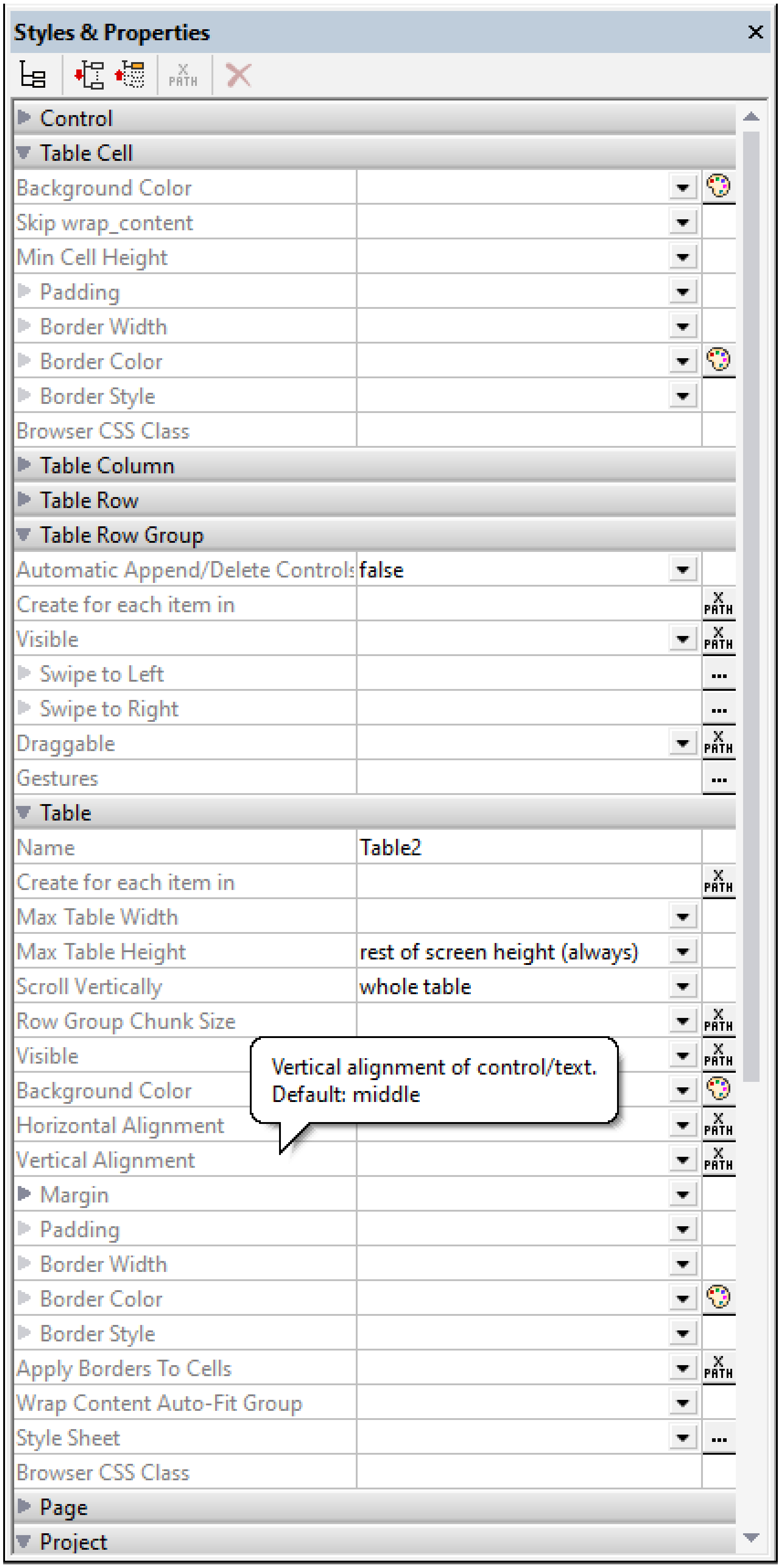
## 12.1.5 Table Properties

Certain properties can be set for a table's cells, columns, and rows, as well as for the entire table. The screenshot below left shows the set of properties that are available for static tables[1063] and repeating tables[1065]. The screenshot below right shows the properties that are available for dynamic tables[1069]. Note that dynamic tables have an additional set of *Table Row Group* properties. This is because a dynamic table typically defines *a group of rows (row group)*, where each row corresponds to a single instance of the same element. For example, if each `Person` child element of an `Office` element corresponds to a row of the table, then all the `Person` elements together constitute the row group.

Table properties are discussed in detail in the description of the Table control[614]. In this section, we discuss those table properties that are unique to MobileTogether or are handled uniquely.



| | |
|---|---|
| ▶ Control | |
| ▼ Table Cell | |
| Background Color | |
| Skip wrap_content | |
| Min Cell Height | |
| ▶ Padding | |
| ▶ Border Width | |
| ▶ Border Color | |
| ▶ Border Style | |
| Browser CSS Class | |
| ▶ Table Column | |
| ▼ Table Row | |
| Visible | |
| Background Color | |
| ▶ Padding | |
| ▶ Border Width | |
| ▶ Border Color | |
| ▶ Border Style | |
| Browser CSS Class | |
| ▼ Table | |
| Name | Table2 |
| Create for each item in | |
| Max Table Width | |
| Max Table Height | rest of screen height (always) |
| Scroll Vertically | whole table |
| Row Group Chunk Size | |
| Visible | |
| Background Color | |
| Horizontal Alignment | |
| Vertical Alignment | |
| ▶ Margin | |
| ▶ Padding | |
| ▶ Border Width | |
| ▶ Border Color | |
| ▶ Border Style | |
| Apply Borders To Cells | |
| Wrap Content Auto-Fit Group | |
| Style Sheet | |
| Browser CSS Class | |
| ▶ Page | |
| ▼ Project | |
| Server Access | |



| | |
|---|---|
| ▶ Control | |
| ▼ Table Cell | |
| Background Color | |
| Skip wrap_content | |
| Min Cell Height | |
| ▶ Padding | |
| ▶ Border Width | |
| ▶ Border Color | |
| ▶ Border Style | |
| Browser CSS Class | |
| ▶ Table Column | |
| ▶ Table Row | |
| ▼ Table Row Group | |
| Automatic Append/Delete Controls | false |
| Create for each item in | |
| Visible | |
| ▶ Swipe to Left | |
| ▶ Swipe to Right | |
| Draggable | |
| Gestures | |
| ▼ Table | |
| Name | Table2 |
| Create for each item in | |
| Max Table Width | |
| Max Table Height | rest of screen height (always) |
| Scroll Vertically | whole table |
| Row Group Chunk Size | |
| Visible | |
| Background Color | |
| Horizontal Alignment | |
| Vertical Alignment | |
| ▶ Margin | |
| ▶ Padding | |
| ▶ Border Width | |
| ▶ Border Color | |
| ▶ Border Style | |
| Apply Borders To Cells | |
| Wrap Content Auto-Fit Group | |
| Style Sheet | |
| Browser CSS Class | |
| ▶ Page | |
| ▼ Project | |

Vertical alignment of control/text.
Default: middle

## Property: Repeating

The **`Repeating`** property is available for repeating tables (the entire table repeats) and static tables. It takes a Boolean value (`true` or `false`) and determines whether the table is repeating[1065] or static[1063]. The property it is not available for dynamic tables[1069].

The value of this property is automatically assigned at the time a repeating or static table is created. A repeating table[1065] has a **`Repeating`** property of `true`, while a static table[1063] has a **`Repeating`** property of `false`. After a table has been created as a particular type (repeating or static), its type can be changed by changing the value of the table's **`Repeating`** property.

## Property: Create for each item in

The **`Create for each item in`** property is available for tables of all types and table row groups (in dynamic tables). It specifies the number of times the repeating table or repeating table row group is re-created. This number is equal to the number of items in the sequence returned by the property's XPath expression. The expression can return two types of sequence:

- Nodes from a page source tree. This is an alternative to associating a repeating table (or table row group) with a page source node—an association made by dragging and dropping the node onto the table[1069]. An XPath expression of this kind also allows more flexibility in the node selection. For example, the XPath expression `$XML1/Offices/Office[@location='US']` returns a sequence of `Office` nodes that have an attribute of `@location='US'`. The US filter cannot be applied by using the alternative method of dropping the `Office` node onto the table. However, this filter can be achieved with the `Create for each item in` property.
- Items unrelated to the page source tree. For example, in the month October 2014, the expression `1 to subsequence(age-details(xs:date("2014-01-01")), 2, 1)` returns a nine-item sequence, namely, the integers from 1 to 9, which is the number of months that have elapsed between 01 January 2014 and a day in October 2014. This is because the basic XPath expression is `1 to X`. And `X` (according to the `subsequence` function) is the second item of the three-item sequence returned by the `age-details` function. This latter function returns the "age" of the current day (in our month of October 2014) relative to the input date (01 January 2014) in terms of the three-item sequence years, months, days (which in this case will be 0 years, 9 months and XX days). The second item of the three-item sequence is the number of months in the age, which is 9. Since the returned sequence contains nine items (the range from `1` to `9`), the table will be created nine times.

**Note:** If you wish to preview the results of XPath expressions, run MobileTogether Designer's built-in simulator (**Run | Simulate Workflow**), and, in the Simulator dialog that appears,click **Evaluate XPath** and then **Evaluator**.

## Enabling row dragging in tables with dynamic rows

A table with repeating rows consists of multiple rows, each of which corresponds to one instance of the repeating data element and which we call a *data row*. For example, a `Person` row group would typically contain multiple `Person` data rows that occur in some specific order. The end user might want to reorder the `Person` data rows and would typically do this by dragging a data row to its new position. In the design, you can enable row dragging and define the actions to perform when the dragged row has been dropped at its new position. You can enable row dragging by using the following properties:

- *Table Row Group* properties: `Draggable` property. If the property is set to `false`, then row dragging is disabled. If the property is set to `true`, then the end user can drag any data row to a new location by first long-tapping or long-clicking anywhere in the data row till Drag mode is activated and then dragging the row to its new location in the table. You can avoid the long-clicking by setting the `Instant Row Dragging`[417] property of any button in the row group to `true`. In this case, the end user can drag immediately. If you set the value of the `Draggable` property to `true`, then a popup appears asking whether you want to automatically let MobileTogether define the reordering actions. Unless you wish to assign other actions *instead* of reordering actions, select **Yes**.

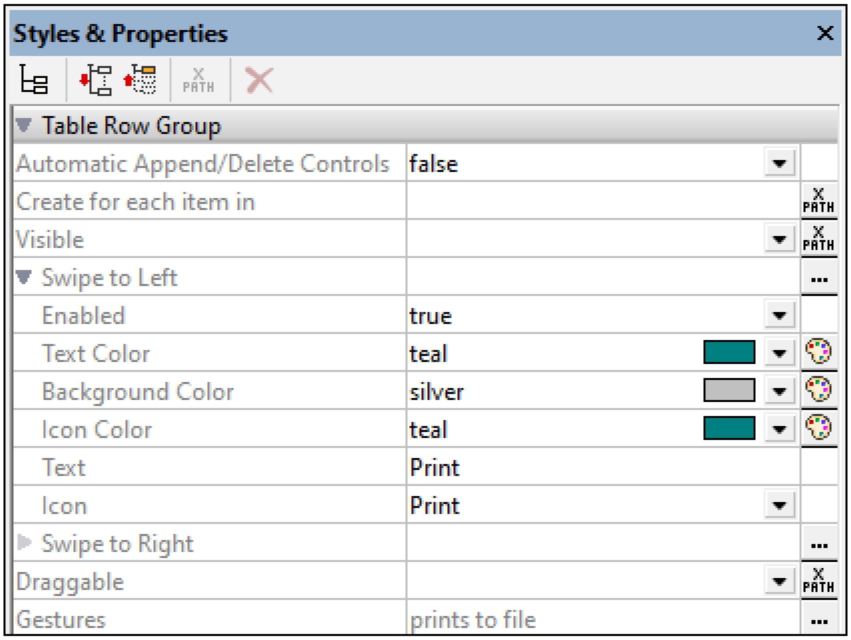
- *Table Row Group* properties: `Gestures` property. This property enables you to set actions to perform when the data row is dropped in its new location or swiped. For example, when a row is dropped at a new location, the reordering is shown in the interface but the underlying data is not automatically reordered. If you want this to happen, actions must be explicitly defined to do this. So, for example, if the end user moves a `Person` row from position 1 to position 5 in the display and you therefore want to move the corresponding `Person` element in the underlying data to the new position, then actions to do this must be specified. You can let MobileTogether define the reordering actions for you automatically (*see previous bullet point*) or you can define the actions yourself. Alternatively, you can let MobileTogether define the reordering actions and then modify or augment these actions.

Note that this feature applies only to the row groups of top-level dynamic tables[1078].

The `Draggable` and `Gestures` properties are described in the Table control[614] topic. The `Instant Row Dragging` property is a property of the Button control[417] and is described with the other Button properties[417].

## Swipe actions for dynamic rows

You can set actions to be performed when the end user left-swipes or right-swipes a data row of a row group. You can set swiping actions by using the `Swipe to Left` and `Swipe to Right` properties of the Table Row Group (*see screenshot below*).



Styles & Properties

| Table Row Group | |
|---|---|
| Automatic Append/Delete Controls | false |
| Create for each item in | |
| Visible | |
| Swipe to Left | |
| Enabled | true |
| Text Color | teal |
| Background Color | silver |
| Icon Color | teal |
| Text | Print |
| Icon | Print |
| Swipe to Right | |
| Draggable | |
| Gestures | prints to file |

Setting up the actions of a Swipe event consists of the following:

- Set the `Enabled` property to `true`.

- On swiping, a text string and an icon can be displayed in the swiped row. Define the corresponding properties according to what you want to display (*see screenshot above*). The icon is selected from a set of available icons.
- Set one or more actions to be carried out on swiping. The Actions[667] dialog for swiping is accessed via the `Gestures` property or the **Additional Dialog** button of the `Swipe to Left/Right` property. In the screenshot above, you can see that a *Print to File* action has been set for the `Swipe to Left` property.

Note that this feature applies only to the row groups of top-level dynamic tables[1078].

These properties are described in the Table control[614] topic.

## Row/Column joining and spanning

To join multiple rows or multiple columns, select, respectively, the row or column in the design that you want to span, and use the appropriate **Join** command from the context menu, the Table menu, or the application toolbar. The selected row/column will be joined with the adjacent row/column you selected. If the joined rows or columns are, respectively, within a row group or column group (which are created for dynamic rows or columns), then the join occurs within each instance of the group, and the joined row/column is displayed in each group.

In the case of dynamic columns, an additional type of column merging, called spanning, is available: The columns of all column groups are spanned to a single column, whether the column group in the design consists of one or more columns. To do this, set the `Spans Column Groups`[614] property (available in the Styles & Properties Pane[274]) to `true`. This property is available only in the first column of a column group. It takes a value of **`true`** or **`false`**. Default is **`false`**. If the property is set to **`true`**, then, in the output, all the columns of the column group are spanned, resulting in one column.

The table below shows a design consisting of a column group comprising two columns, which are not spanned. The column group is associated with the element **`Node`**.

*Column Group in design, corresponds to the repeating element* **`Node`**. ***Not spanned***

| | |
|---|---|
| Column-1 in design | Column-2 in design |

In the output, the column group is repeated for each instance of **`Node`**. As a result, two columns are created for each **`Node`** element, as shown in the table below.

| | | | | | | |
|---|---|---|---|---|---|---|
| **`Node[1]`**<br>*Column-1* | **`Node[1]`**<br>*Column-2* | **`Node[2]`**<br>*Column-1* | **`Node[2]`**<br>*Column-2* | ... | **`Node[n]`**<br>*Column-1* | **`Node[n]`**<br>*Column-2* |

If the dynamic column group were spanned, by setting the `Spans Column Groups`[614] property to `true`, then the effect in the design is as if both columns were joined (*see table below*). The properties and content of the resulting column will be that of the first column.

*Column Group in design corresponding to repeating element* **`Node`**: ***spanned***

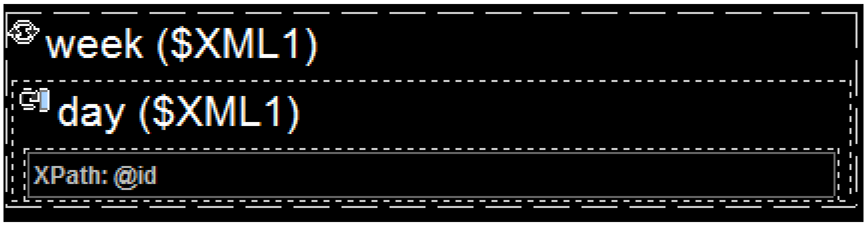
Column-1 in design subsumes Column-2

In the output, the column group is spanned across all instances of **`Node`**. As a result, there will be only one column for all the instances of **`Node`**, as shown in the table below. If the content of the column is dynamically selected via an XPath expression that locates **`Node`** elements, then an error will be returned.

**`Node[1 to n]`**
*Column-1 spans to cover all instances of* **`Node`**

When dynamic columns are spanned, you can think of the transformation process as occurring in two steps: (i) In the design, all columns (of any type, including static) in the column group are joined together into one column, as if the **Join** 1087 command 1087 were used on them; (ii) in the output, all the instances of the repeating element are created as a single column. Any XPath expression that: (i) is located within a spanned dynamic column in the design, and (ii) tries to locate individual element instances corresponding to output columns will return an error.

The screenshot below shows a simple example of a dynamic column created in a column group. The column group in the design contains a single column group that is associated with the **`day`** element, and this column group is within a (repeating) table 1065 associated with the **`week`** element (which, in the page source, is the parent of the **`day`** element). Since the **`week`** element repeats, a new table will be created for each **`week`** element. If, in the page source, there are multiple **`day`** child elements of the **`week`** element, and if, in the design, the dynamic columns of the column group are not spanned, then the table (for each **`week`**) generated from this design will have as many columns as there are **`day`** child elements. If, however, you set the `Spans Column Groups` 614 property to `true`, then the columns in the generated table will be spanned, and the table will have only one column.

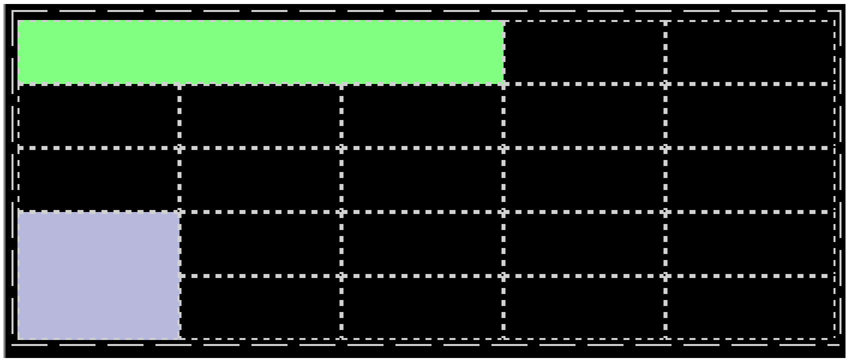


For more information about column groups, see Dynamic Columns 1074.

## Column/Row visibility

The visibility of a column or row is set by selecting the column or row, and setting its **`Visible`** property to **`true`** or **`false`**. The default value is **`true`**.

If columns or rows are spanned, the visibility of the spanned columns/rows can be specified individually **`if`** the visibility of the first column/row in the spanned set has been given a value of **`true`**. If the visibility of the first column/row in the spanned set of columns/rows has been given a value of **`false`**, then all the columns/rows in the spanned set are given a value of **`false`**.



*Screenshot above: Spanned columns in the first row (colored green):*

- The three spanned columns in Row-1 (green) are considered to be Column-1. The next column in Row-1 is Column-4. In Row-1, there is no Column-2 or Column-3.
- If you select any of the first three columns individually in Rows 2 to 5 and set the visibility to `true/false`, then the visibility of none of the other columns will be affected.
- If you select (the spanned) Column-1 in Row-1 and set the visibility to `true/false`, then the visibility of only Column-1 will be modified. Column-2 and Column-3 (in Rows 2 to 5) will not be affected.

*Screenshot above: Spanned rows in the first column (colored blue):*

- The spanned rows in Column-1 (blue) are considered to be Row 4. In Column-1, there is no Row-5.
- If you select either of the two rows (Row-4 or Row-5) individually in Columns 2 to 5 and set the visibility to `true/false`, then the other row will not be affected.
- If you select (the spanned) Row-4 in Column-1 and set the visibility to `true/false`, then the visibility of only Row-4 will be modified. Row-5 (in Columns 2 to 5) will not be affected

## Scrollable tables

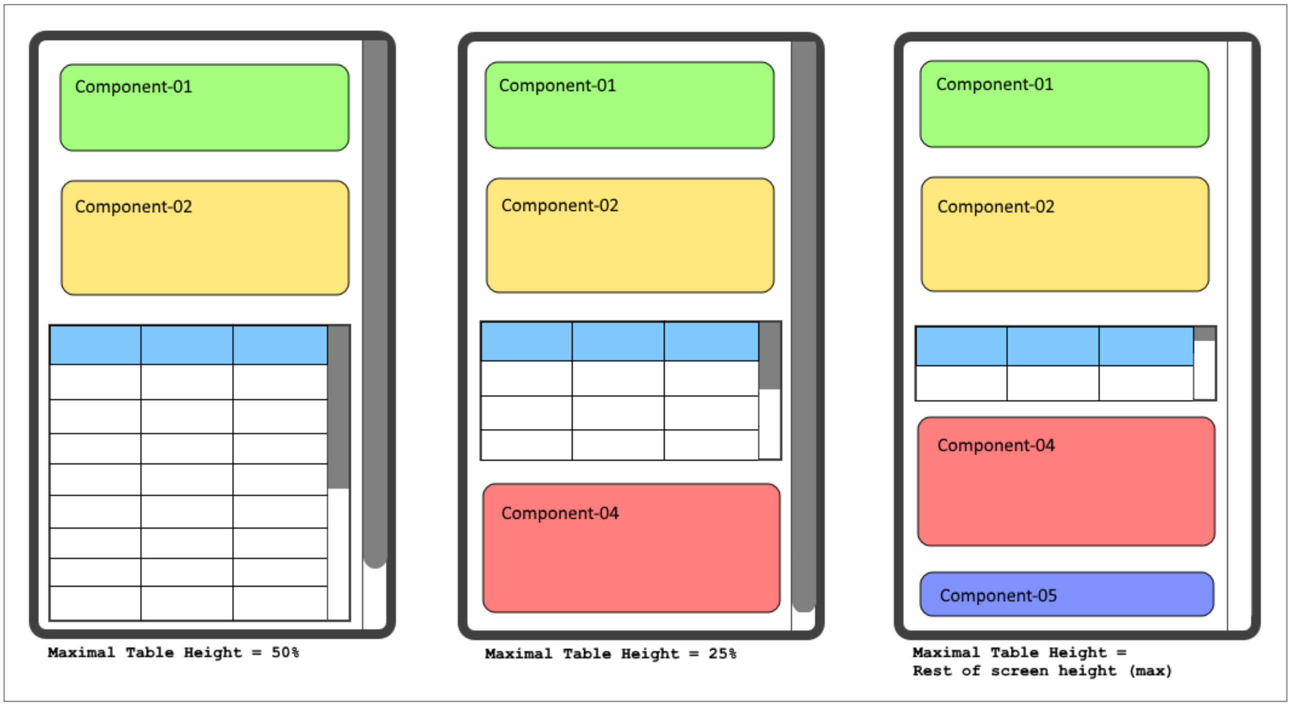
If a table is very long and/or very wide, you can set it to be vertically and/or horizontally scrollable. In this event, only a certain portion of the table will be displayed; the rest can be scrolled in and out of view.

*Vertical scrolling*
The **`Max Table Height`** 614 property specifies the table's height, in pixels or relative to the device's screen height. Select one of the values from the property's combo box. For example, if you select `50%`, then the table will have a height of half of the device's screen height (*see screen at first left in image below*). If the table has a vertical extent that does not fit in the allotted screen space, then the table will have a scroll bar and the end user can scroll the rest of the table in and out of the allotted screen space (in this example, 50% of the screen height). If design components occur above the table, then all these components are displayed above the table; the table itself will have the absolute or relative height specified via this property.

**Note:** The table and the page have separate scroll bars (*see screens in the image below*). In the MobileTogether Designer Simulator, use the scroll wheel to scroll vertically, and click-and-drag to scroll horizontally.

**Note:** On **Android 4.x** devices, if there are two or more (scrollable or non-scrollable) tables on a page, then, if any of these tables is scrollable, it cannot be scrolled vertically.

**Note:** See the Scrollable Tables tutorial 236 for an example.



The **`Max Table Height`** 614 property can take two other values (besides a pixel or percentage value):

- *Rest of screen height (max):* The table height is minimized as much as possible so that as much of the rest of the page can be displayed. In the image above, the screen at extreme right shows a table that has been given this property value: The table height has been reduced so that all the five components of the page are displayed. Notice, in this case, that the scroll bar of the page has been reduced to zero since the entire page is in view.
- *Rest of screen height (always):* This option enables you to use the entire screen height to display the page. If a table does not have enough vertical extent to fill the page, then additional space is added below the table so that the last component of the page is displayed just above the bottom of the screen. This setting allows you to constrain end-of-page content to a location at the bottom of the screen.

The **`Scroll Vertically`** 614 property becomes available after a value has been set for the **`Max Table Height`** 614 property **and** no **`Max Table Width`** 614 value is set. The **`Scroll Vertically`** 614 property can take one of two values:

- *Whole table:* The whole table scrolls in and out of the table height that is allotted to the table via the **`Max Table Height`** 614 property. *Whole table* is the default value.
- *Rows except header and footer:* The table's header and footer stay fixed in the view. The table's body rows scroll within the remaining table height.

The **`Row Group Chunk Size`** 614 property becomes available only if there is a repeating row-group in the table and after a value has been set for the **`Max Table Height`** 614 property of scrollable tables. It enables you to specify the number of row groups that are loaded at a time. When the user scrolls down and the last row group of the last-loaded chunk is reached, then the next chunk is loaded. There is no default value for this property.

*Horizontal scrolling*

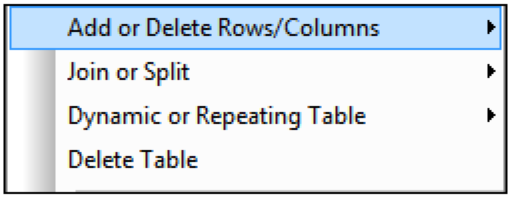
The **Max Table Width** 614 property specifies the table's width: (i) in pixels, (ii) relative to the device's screen width, or (iii) optimized for columns (`wrap_content`). The default is `wrap_content`. Select the value you want from the dropdown list of the property's combo box. If the table width is larger than the screen width, then the table will be displayed with a horizontal scroll bar. Users can swipe left or right to scroll the table.

### Table borders

The Table 1637 menu offers a number of commands to edit table structure. Several table properties are set in the Styles & Properties Pane 274. Additionally, table borders can be conveniently set in the Border Settings dialog (**Table | Border Settings** 1644).

## 12.1.6 Table Context Menu

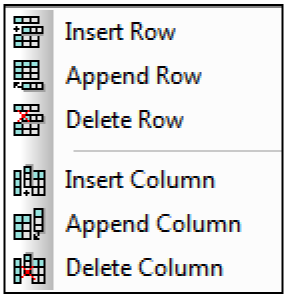
The context menu of all tables (static 1063, repeating 1065, and tables with dynamic rows 1069 and/or dynamic columns 1074) has the same commands (*see screenshot below*). These commands allow the structure of the table to be modified after the table has been created.



### Add or Delete Rows/Columns

Hovering over the **Add or Delete Rows/Columns** command displays a submenu (*screenshot below*) with commands that allow you to insert/append rows/columns relative to the currently selected cell. Note that rows and columns added in this way are static. This means, for example, that if one static row is added in the design, then it will result in one static row in the output. Of course, if the row is added within a repeating structure, then the static row will also repeat.

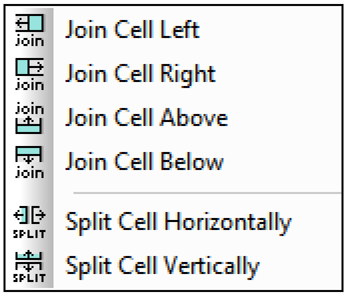
The currently selected row/column can also be deleted.



These commands are available when a row or column of any kind of table (static 1063, repeating 1065, or tables with dynamic rows 1069 and/or dynamic columns 1074) is selected.

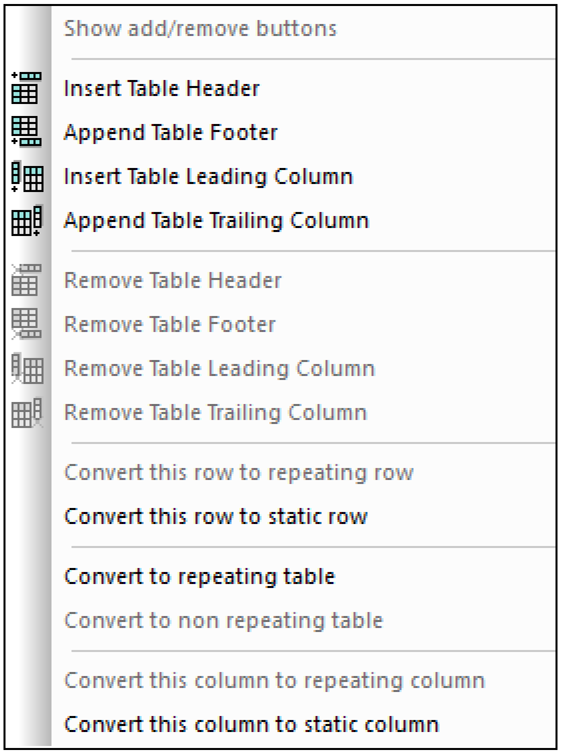
## Join or Split

Hovering over the **Join or Split** command displays a submenu (*screenshot below*) with commands that allow you to join the currently selected cell with an adjacent cell. The currently selected cell can also be split horizontally or vertically. These commands are available for the cells of all tables (static[1063], repeating[1065], and tables with dynamic rows[1069] and/or dynamic columns[1074]).



## Dynamic or Repeating Table

Hovering over the **Dynamic or Repeating Table** command displays a submenu (*screenshot below*).


Show add/remove buttons
Insert Table Header
Append Table Footer
Insert Table Leading Column
Append Table Trailing Column
Remove Table Header
Remove Table Footer
Remove Table Leading Column
Remove Table Trailing Column
Convert this row to repeating row
Convert this row to static row
Convert to repeating table
Convert to non repeating table
Convert this column to repeating column
Convert this column to static column

This submenu contains commands that allow you to do the following:

- Specify that Append/Delete controls are added automatically to the rendered table (only in repeating[1065] tables and tables with dynamic rows[1069]).

- Insert/remove a header/footer (in tables with dynamic rows[1069]), and insert/remove a leading/trailing column (in tables with dynamic columns[1074]).
- If the selected row ***is not*** a dynamic repeating row, convert it to a repeating row. You can also convert static headers and footers separately to repeating rows. When a table has been converted to a repeating row, you can add a header and footer to it.
- If the selected row ***is*** a repeating row, convert it to a static row.
- If the selected column ***is not*** a repeating column, convert it to a repeating column.
- If the selected column ***is*** a repeating column, convert it to a static column.
- Convert the table type between repeating[1065] and static[1063]. (Note that the rows and columns of a static[1063] table can be converted, respectively, to repeating rows and repeating columns).

## Delete Table

Deletes the selected table.

# 12.2 Images

Images can be added to the design by both the designer and the end-user, and images can be added via an URL or can be stored as Base64-encoded text in XML files. The basis of image functionality is provided by the Image control[541], which positions the image in the design and defines the basic properties of the image. Key mechanisms and usage are described in the sub-sections of this section.

- Image Source[1090] is the property of the Image control[541] that selects the image to display. This section describes the two types of image sources that can be used: (i) image files located by URL, and (ii) images as Base64-encoded strings.
- Image Size[1091]
- Base64-Encoded Images[1098] describes how to use Base64-encoded images in your design.
- Exchangeable Image File Format (Exif)[1092] is a format for storing image metadata with an image. This section shows how individual pieces of Exif data can be retrieved and used in a design.
- Images Chosen by the End User[1101] explains the mechanism with which the end user of a MobileTogether solution can select images that will be stored in a database. These images can be stored as image files or as Base64-encoded strings.
- Transforming Images[1106] describes how to transform Base64 images (for example, resizing or rotating an image) and the issues related to transforming (loss of Exif data and memory issues on the client).
- Images in Databases[1107] lists the ways in which images can be stored in databases.

These mechanisms are enabled by powerful Image actions[704] and image-related Altova XPath extension functions[1718].

## 12.2.1 Image Source

The following types of image sources can be used in page designs:

- Binary image files of common formats, such as `PNG`, `BMP`, `JPG`. Images that have binary file sources reference the URL of the image file.
- Base64-encoded strings, which are text encodings of images. Images that have Base64-encoded images access the Base64-encoded string via an XPath expression. The XPath expression typically returns a node containing the Base64 string. MobileTogether reads the Base64 string and generates the encoded image from the string.

### Inserting an image in the design

To insert an image in the design, do the following:

1. Drop an Image control[541] into the design.
2. In the Styles & Properties Pane[274], set the image property **`Image Source Type`**[541] to either `url` or `base64`, to match the type of the image being inserted. The default setting of this property is `url`.
3. Specify the image in the **`Image Source`**[541] property. If an image file is being referenced, specify the URL. If a Base64-encoded image is being referenced, use the **`Image Source`**[541] property's XPath expression either to supply the Base64 string directly or to supply the XML node containing the Base64 string. Note that, for both source types (`url` or `base64`), there are two alternative ways of specifying the property's value: (i) With the **`Image Source`**[541] property selected, click the XPath toolbar button of the Page Sources Pane[270], and enter an XPath expression that evaluates to the URL

or the Base64 string; (ii) Drop an XML node containing the URL or Base64 string from the Page Sources Pane[270] onto the Image control[541].

**Note:** Every time an image source is changed (for example, by a user selection), a Reload action[801] for the image (unless it is a Base64 image) is required in order to display the new image.

### Inserting image files via URL

Insert the image file by browsing for it or selecting a global resource. For details, see the **`Image Source`**[541] property. For an example of inserting images via a URL, see the QuickStart Tutorial[83].

*Embedding URL-sourced images in the design file*
If an image is sourced via a URL (and not as a Base64-encoded image), then the image can be embedded in the design file. Use the **`Embed Image`**[541] property of the Image control[541] to do this. If this property is set to true, the image is converted to Base64-encoding and embedded in the design file. If the original image is modified, you can re-embed via the Image control's context menu. You can also save the embedded image to a file via the context menu.

This setting can be automatically set to `true` if, when selecting an image, the option to embed the image is checked. See the **`Image Source`**[541] property.

### Inserting Base64-encoded images

When an image is encoded as Base64 text, it can be stored as the text content of an XML element node. As a result it is easier to transport, and its metadata can be easily parsed and retrieved. In the listing below, the Base64-encoded image is the content of the `<png>` element.

```
<images><png>iVBORw0KGgoAAAANSUhEU...</png></images>
```

To insert a Base64-encoded image, the XPath expression of the **`Image Source`**[541] property must resolve to the image's Base64-encoded text string. You can also drop an XML node that contains the image's Base64-encoded text string from the Page Sources Pane[270] onto the Image control[541].

See the next section, Base64-Encoded Images[1098], for an example of how to use Base64-encoded images.

## 12.2.2 Image Size

An image control which is placed in the design as a top-level control (that is, directly on the page) has a **`Control Width`** property, but an image placed inside a table cell has both a **`Control Width`** property as well as a **`Control Height`** property. This enables the following image size settings:

- *Control Width only:* (i) `wrap_content`: the image is displayed at its original size; `fill_parent`: the image fills the width of the page.
- *Control Width and Control Height:* (i) If both are set to `fill_parent`, then the image is set to the height of the row and the width is scaled to maintain the image's aspect ratio or AR (which is the image's height/width ratio); (ii) In the three other combinations of these two settings setting, the image is set to the height of the image. The white cells in the table below give the respective image size at the intersection of the properties in the respective row and column.

| | *Control Height* = `wrap_content` | *Control Height* = `fill_parent` |
|---|---|---|
| *Control Width* = `wrap_content` | Image height; width scaled to keep AR | Image height; width scaled to keep AR |
| *Control Width* = `fill_parent` | Image height; width scaled to keep AR | Height of row; width scaled to keep AR |

## 12.2.3 Exchangeable Image File Format (Exif)

**Exchangeable image file format (Exif)** is a standard that defines the image formats used by some digital cameras and smartphone cameras. The metadata tags of  the Exif standard hold a broad range of information ranging from the image's date-time and geolocation to camera settings and image composition details. When an Exif image is converted to Base64-encoding, the metadata in the image is also converted to Base64, and is available for retrieval.

**Note:** Not all digital cameras or smartphone cameras provide Exif data.

### MobileTogether Designer's Exif functionality

MobileTogether Designer provides the following Exif-related functionality:

- The Let User Choose Image action[706] provides an option that starts the camera application on the end user's client device. The photo that is taken is saved in an XML node as a Base64-encoded image. If the camera application uses the Exif format, then Exif metadata is also saved in the Base64-encoded image. This data is available for immediate retrieval from the XML node.
- An Altova XPath extension function called **`image-exif-data`**[1718] takes a Base64-encoded JPEG image as its argument and returns the Exif metadata contained in the string as attribute-value pairs. (See the description of the **`image-exif-data`**[1718] function for details. To find just the dimensions of images, use the MobileTogether XPath extension function **`mt-image-width-and height`**[1262].)
- An Altova XPath extension function called **`suggested-image-file-extension`**[1718] takes a Base64 string as its argument, and returns an image file extension (such as `jpg`, `png`, `bmp`). This is useful for automatically detecting the correct image format and saving the file with an appropriate file extension.
- The Load/Save Image to File action[707] enables a Base64-encoded image to be saved in a binary image format (such as `jpg`, `png`, `bmp`). Exif data is saved in the Base64-encoded text.

The example below explains how Exif data can be retrieved from a Base64-encoded image, and how this data can be used in a solution.

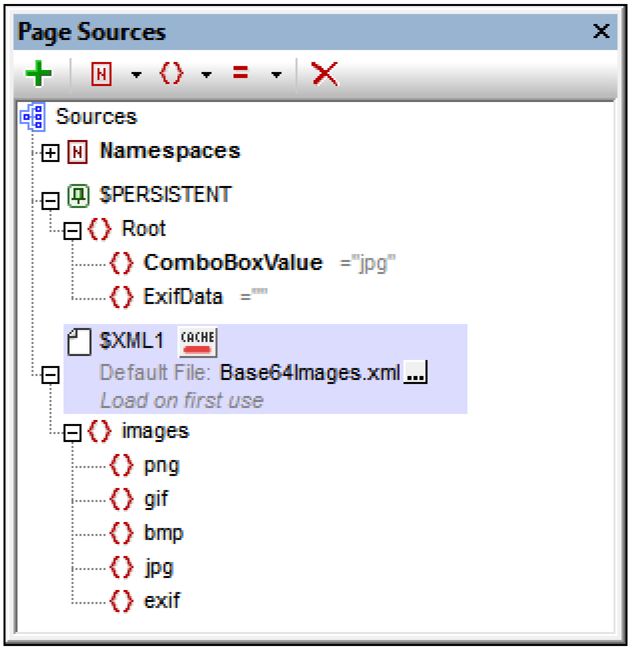
**Note:** Exif data[1092] will be lost if the image is resized[1106] or rotated.

### Example file: Base64Images.mtd

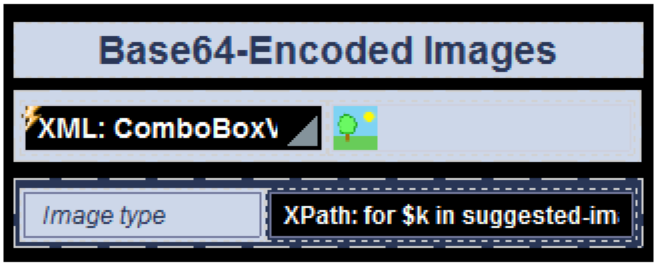
The design file `Base64Images.mtd` is located in your *([72]My) Documents*[72] MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\Images`**. You can open this file in MobileTogether Designer, run it in the simulator (**F5**), and look through the design definitions. The design's default file contains one image with Exif metadata.

## ⊟ *Basic design*

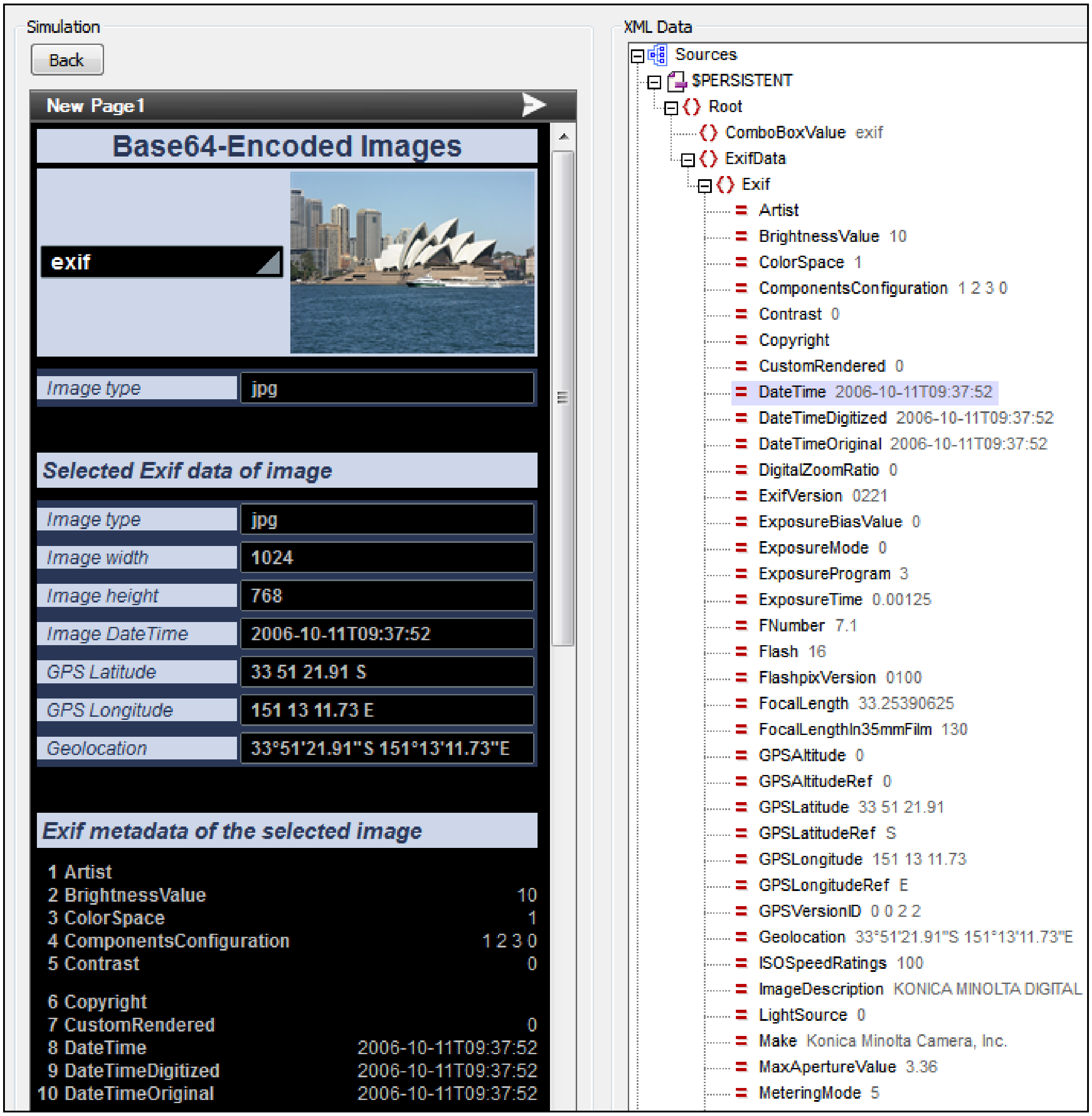
The design file uses Base64-encoded images that are stored in the XML file `Base64Images.xml` (also located in the `Tutorials\Images` folder). The structure of the XML file is shown in the screenshot below. The `images` element has six child elements, each of which has an image of a different format stored as a Base64 text string. It contains one image with Exif metadata (the `exif` element). The `$PERSISTENT` tree is used to save temporary user selections (`ComboBoxValue`) and Exif metadata (`ExifData`).



The top part of the design (*screenshot below*) has a label for the page title, and two tables. The design of this part of the page is described in the previous section, Base64-Encoded Images 1098. The objective is to allow the end user to select an image type in the combo box. This selection determines which Base64-encoded image in the XML file is selected for display in the cell to the right-of the combo box.



If the user selects the **`exif`** item in the combo box, then the Base64-encoded image in the `exif` element of the XML file is displayed. Exif metadata is displayed in two tables (*"Selected Exif data of image"* and *"Exif metadata of the selected image"; see simulator screenshot below*). In the simulator, if you expand the `$PERSISTENT` tree in the XML Data pane (*see screenshot below*), you will see the Exif data that has been retrieved from the Base64 string. The design of the two Exif data display tables is described below. See the previous section, Base64-Encoded Images 1098, for a detailed description of other parts of the design.



Base64-Encoded Images

exif

| | |
|---|---|
| Image type | jpg |

Selected Exif data of image

| | |
|---|---|
| Image type | jpg |
| Image width | 1024 |
| Image height | 768 |
| Image DateTime | 2006-10-11T09:37:52 |
| GPS Latitude | 33 51 21.91 S |
| GPS Longitude | 151 13 11.73 E |
| Geolocation | 33°51'21.91"S 151°13'11.73"E |

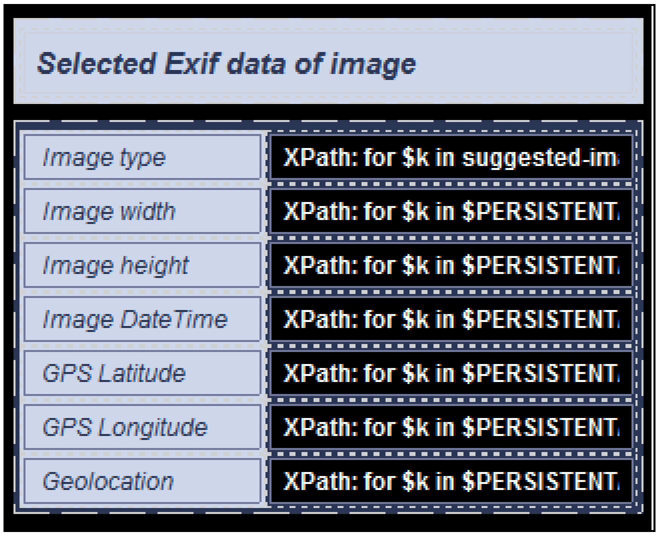
Exif metadata of the selected image

| | |
|---|---|
| 1 Artist | |
| 2 BrightnessValue | 10 |
| 3 ColorSpace | 1 |
| 4 ComponentsConfiguration | 1 2 3 0 |
| 5 Contrast | 0 |
| 6 Copyright | |
| 7 CustomRendered | 0 |
| 8 DateTime | 2006-10-11T09:37:52 |
| 9 DateTimeDigitized | 2006-10-11T09:37:52 |
| 10 DateTimeOriginal | 2006-10-11T09:37:52 |

```
Sources
  $PERSISTENT
    Root
      ComboBoxValue  exif
      ExifData
        Exif
          Artist
          BrightnessValue  10
          ColorSpace  1
          ComponentsConfiguration  1 2 3 0
          Contrast  0
          Copyright
          CustomRendered  0
          DateTime  2006-10-11T09:37:52
          DateTimeDigitized  2006-10-11T09:37:52
          DateTimeOriginal  2006-10-11T09:37:52
          DigitalZoomRatio  0
          ExifVersion  0221
          ExposureBiasValue  0
          ExposureMode  0
          ExposureProgram  3
          ExposureTime  0.00125
          FNumber  7.1
          Flash  16
          FlashpixVersion  0100
          FocalLength  33.25390625
          FocalLengthIn35mmFilm  130
          GPSAltitude  0
          GPSAltitudeRef  0
          GPSLatitude  33 51 21.91
          GPSLatitudeRef  S
          GPSLongitude  151 13 11.73
          GPSLongitudeRef  E
          GPSVersionID  0 0 2 2
          Geolocation  33°51'21.91"S 151°13'11.73"E
          ISOSpeedRatings  100
          ImageDescription  KONICA MINOLTA DIGITAL
          LightSource  0
          Make  Konica Minolta Camera, Inc.
          MaxApertureValue  3.36
          MeteringMode  5
```

### ⊟ Selected Exif data of image

- Selected Exif data is presented in a static table consisting of two columns and multiple rows (*screenshot below*).
- The first column contains labels; the second column contains edit boxes, each having an XPath expression that returns one piece of Exif metadata.



Selected Exif data of image

| | |
|---|---|
| *Image type* | XPath: for $k in suggested-im |
| *Image width* | XPath: for $k in $PERSISTENT. |
| *Image height* | XPath: for $k in $PERSISTENT. |
| *Image DateTime* | XPath: for $k in $PERSISTENT. |
| *GPS Latitude* | XPath: for $k in $PERSISTENT. |
| *GPS Longitude* | XPath: for $k in $PERSISTENT. |
| *Geolocation* | XPath: for $k in $PERSISTENT. |

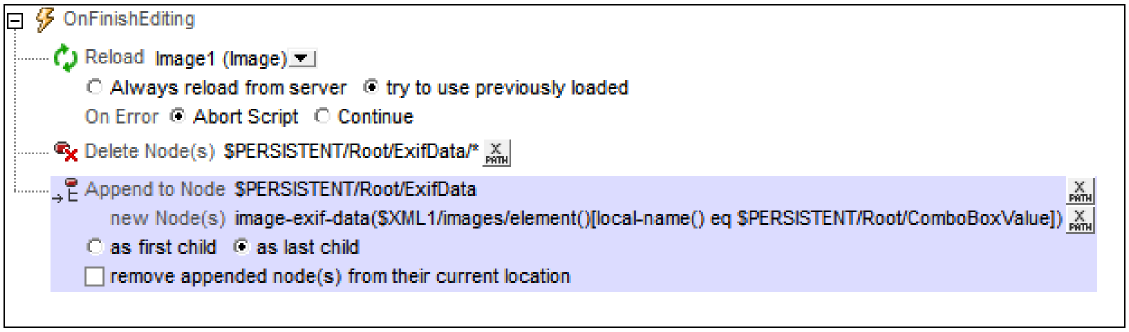
- The *Image type* information is obtained from the Base64-encoded text string by using the Altova XPath extension function `suggested-image-file-extension`[1718]. This function takes a string (the Base64 image) as its argument and retrieves the file-extension information from the string. If no file-extension information is available in the Base64 string, then the function returns the empty string. The XPath expression used is:

  ```
  for $k in suggested-image-file-extension($XML1/images/element()[local-name() eq
  $PERSISTENT/Root/ComboBoxValue])
  return if ($k != '') then $k else "Data not available"
  ```

  The expression provides alternative return strings depending on whether the function returns a non-empty string or an empty string. If a non-empty string is returned, then the string is displayed; if an empty string is returned, an appropriate message is displayed.
- All the other XPath expressions in the table (besides the first row) use the Altova XPath extension function `image-exif-data`[1718] to obtain one piece of Exif metadata. This function takes a string (the Base64 image) as its argument and returns an element node (named **`Exif`**) with attributes holding the Exif metadata. Each attribute-value pair corresponds to one Exif metadata tag. In the expression below, the function `image-exif-data`[1718] returns the **`Exif`** element with multiple attributes. The metadata information we want to obtain with this expression is the width of the image. This information is stored in the `@PixelXDimension` attribute node of the `Exif` element.

  ```
  for $k in $PERSISTENT/Root/ExifData/Exif
  return if ($k/@PixelXDimension !='') then $k/@PixelXDimension else "Data not
  available"
  ```

  The expression checks whether the **`Exif/@PixelXDimension`** node is non-empty or empty. If it is non-empty, then the string is displayed; otherwise, an appropriate message is displayed.
- For more information about the `image-exif-data`[1718] function, see its description in the Altova extension functions section[1718].
- Note the last *Geolocation* value in the table. It is obtained via an Altova-created `Exif/@Geolocation` attribute.
- The `$PERSISTENT/Root/ExifData` node is populated with the Exif data by appending a child node to it that contains the result of the `image-exif-data`[1718] function. This is done by specifying an Append Node action[875] on the combo box that selects which image to display (*see screenshot below*). The action is triggered by the `OnFinishEditing` event of the combo box.

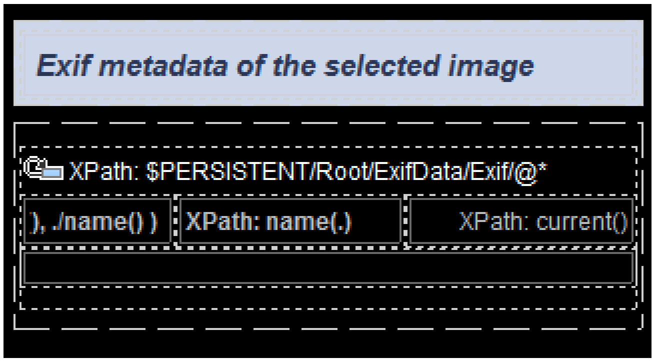


```
OnFinishEditing
    Reload  Image1 (Image)
        ○ Always reload from server  ● try to use previously loaded
        On Error  ● Abort Script  ○ Continue
    Delete Node(s)  $PERSISTENT/Root/ExifData/*
    Append to Node  $PERSISTENT/Root/ExifData
        new Node(s)  image-exif-data($XML1/images/element()[local-name() eq $PERSISTENT/Root/ComboBoxValue])
        ○ as first child  ● as last child
        □ remove appended node(s) from their current location
```

Note the following points:
(i) The XPath locator expression in the Append Node action[875] locates the node in the XML file that has the same name as the string in `$PERSISTENT/Root/ComboBoxValue`.
(ii) The `$PERSISTENT/Root/ExifData` node is deleted before the Exif data (in the `Exif` node) is appended to `ExifData`.
(iii) A page action has been set to delete the `Exif` node in the `$PERSISTENT` tree. This prevents a possible incongruence between the initial image (`jpg`) and old Exif data in the `ExifData` node.

### *All Exif data of image*

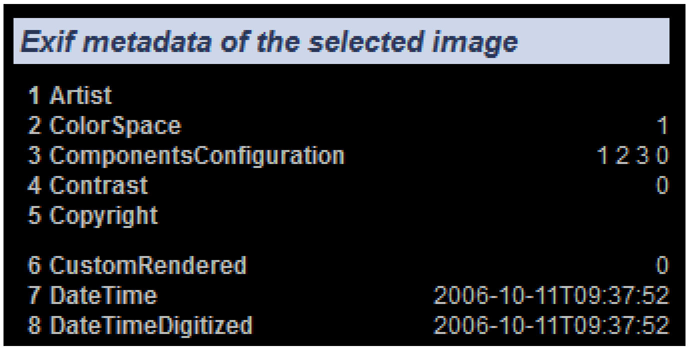
- A table with a repeating row (*screenshot of design below left; simulator view below right*) is used to display all the attribute-value pairs returned by the `image-exif-data`[1718] function.
- The repeating row is specified with an XPath expression that selects all the attributes of the **`Exif`** element node returned by the `image-exif-data`[1718] function: `$PERSISTENT/Root/ExifData/Exif/@*`.



Exif metadata of the selected image

XPath: $PERSISTENT/Root/ExifData/Exif/@*

| | | |
|---|---|---|
| ), ./name() ) | XPath: name(.) | XPath: current() |



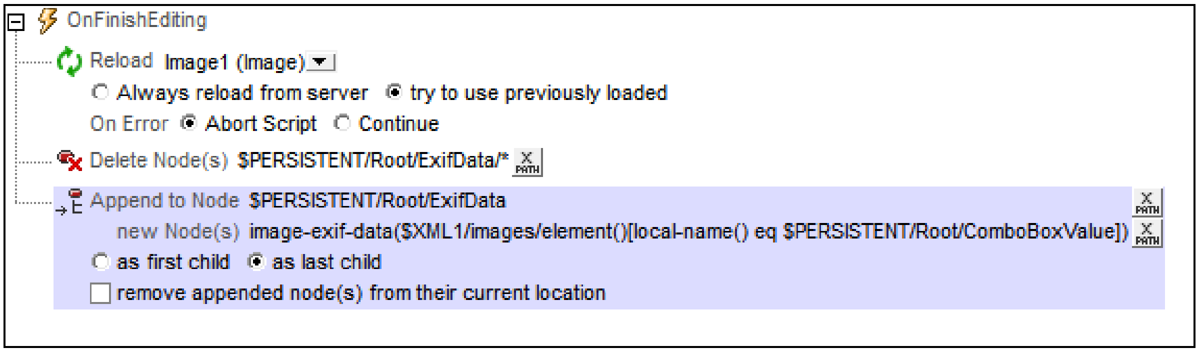
Exif metadata of the selected image

| | | |
|---|---|---|
| 1 | Artist | |
| 2 | ColorSpace | 1 |
| 3 | ComponentsConfiguration | 1 2 3 0 |
| 4 | Contrast | 0 |
| 5 | Copyright | |
| 6 | CustomRendered | 0 |
| 7 | DateTime | 2006-10-11T09:37:52 |
| 8 | DateTimeDigitized | 2006-10-11T09:37:52 |

- The table's first column contains the index position of the current attribute: `index-of(../@*/name(), ./name() )`
- The second column contains the name of the current attribute: `name(.)`
- The third column contains the value of the current attribute: `current()`
- Not all images contain the same Exif metadata. In some cases, some metadata may be absent; in other cases, additional metadata might be present; in still other cases, metadata might be tagged with non-standard, vendor-specific tags. Consequently, knowing what metadata is available and under what attribute names is important. Only with this information can specific attribute values be retrieved.
- If we know the names of the attributes that are returned, we can access its value by using the `image-exif-data` [1718] function like this: **`image-exif-data(Base64String)/@WantedAttribute`**. Note that the function returns the **`Exif`** element.

## *Populating the $PERSISTENT tree with Exif data*

- It can be useful to see all the attribute-value pairs returned by the `image-exif-data` [1718] function. In order to display attribute-value pairs, we can store this output conveniently in the temporary `$PERSISTENT` tree.
- In our example design, the `$PERSISTENT/Root/ExifData` node is populated with the Exif data by appending a child node to the `ExifData` node that contains the result of the `image-exif-data` [1718] function.
- This is done by specifying an Append Node action [875] on the combo box that selects which image to display (*see screenshot below*). The Append Node action [875] is triggered by the `OnFinishEditing` event of the combo box.



OnFinishEditing

Reload Image1 (Image)

○ Always reload from server ● try to use previously loaded

On Error ● Abort Script ○ Continue

Delete Node(s) $PERSISTENT/Root/ExifData/*

Append to Node $PERSISTENT/Root/ExifData

new Node(s) image-exif-data($XML1/images/element()[local-name() eq $PERSISTENT/Root/ComboBoxValue])

○ as first child ● as last child

□ remove appended node(s) from their current location

- The XPath locator expression in the Append Node action 875 locates the node in the XML file that has the same name as the string in `$PERSISTENT/Root/ComboBoxValue`.
- The `$PERSISTENT/Root/ExifData` node is deleted before the Exif data returned by the `image-exif-data` 1718 function is appended to the `ExifData` node.
- If we know the names of the attributes that are returned, we can access the value of any attribute by using the `image-exif-data` 1718 function like this: **`image-exif-data(Base64String)/@WantedAttribute`**. Note that the function returns the **`Exif`** element.

## 12.2.3.1 Base64-Encoded Images

When an image is encoded as Base64 text, it can be stored as the text content of an XML element node. In the listing below, the Base64-encoded image is the content of the `<png>` element.

```
<images><png>iVBORw0KGgoAAAANSUhEU...</png></images>
```

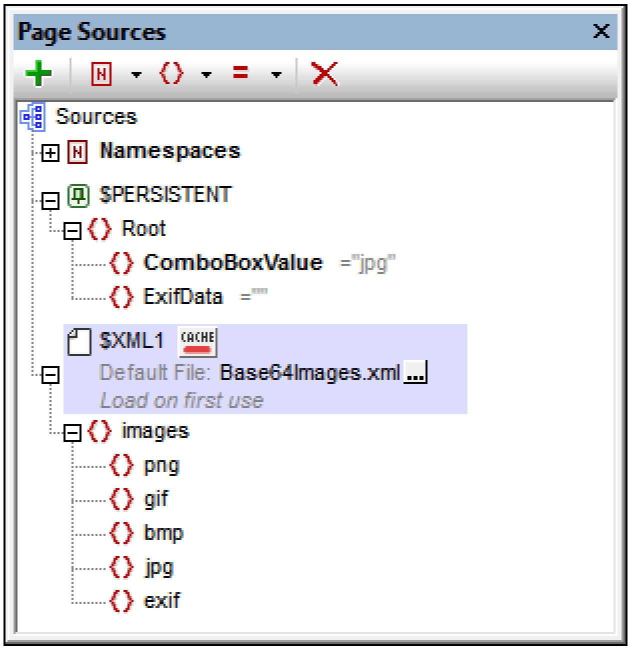
To insert a Base64-encoded image, the XPath expression of the Image control's 541 **`Image Source`** 541 property must resolve to the image's Base64-encoded text string. You can also drop an XML node that contains the image's Base64-encoded string from the Page Sources Pane 270 onto the Image control 541. The example below explains how Base64 images can be used in designs.

**Note:** Images in page source nodes and in databases are stored as Base64-encoded images.

### Example file: Base64Images.mtd

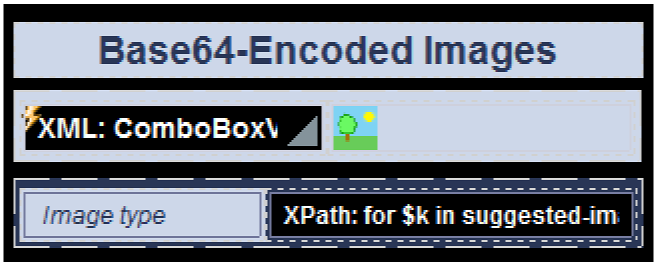
The design file `Base64Images.mtd` is located in your *( 72 My) Documents* 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\Images`**. You can open this file in MobileTogether Designer, run it in the simulator (**F5**), and look through the design definitions.

The design file uses Base64-encoded images that are stored in the XML file `Base64Images.xml` (also located in the `Tutorials\Images` folder). The structure of the XML file is shown in the screenshot below. The `images` element has five child elements, each of which has an image of a different format stored as a Base64 text string. The `$PERSISTENT` tree is used to save temporary user selections (`ComboBoxValue`) and the Exif data of the selected image, if such data exists.


Page Sources

Sources
- Namespaces
- $PERSISTENT
  - Root
    - ComboBoxValue ="jpg"
    - ExifData =""
- $XML1
  Default File: Base64Images.xml
  Load on first use
  - images
    - png
    - gif
    - bmp
    - jpg
    - exif

The design (*screenshot below*) has a label for the page title, and two tables. The first table contains a combo box and an image. The second table contains a label and an edit field.

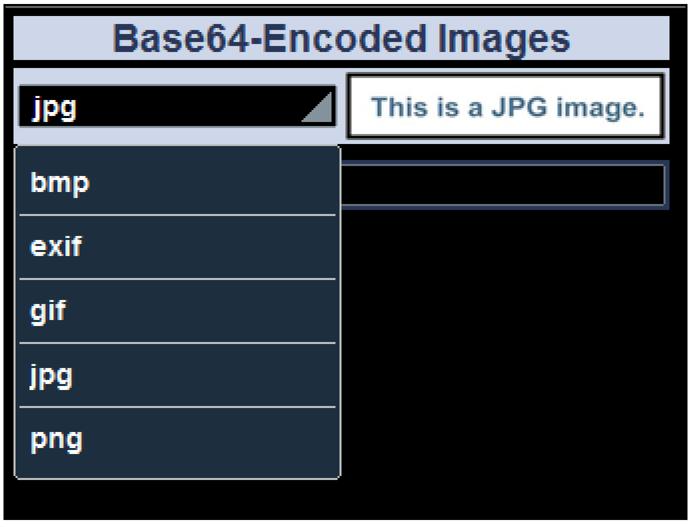

Base64-Encoded Images

XML: ComboBoxV

Image type

XPath: for $k in suggested-im

What we want to do is to select an image type in the combo box (*see simulator screenshot below*). The combo box selection is used to select the Base64 image (from the XML file) to display.



Here are the important points to note:

- The `OnPageLoad`[389] event initializes the `$PERSISTENT/ComboBoxValue` node with a value of `jpg`.
- The combo box is associated with the node `$PERSISTENT/ComboBoxValue` (done by dropping the node from the Page Sources Pane[270] onto the combo box). This association means that the current value of the node is displayed in the combo box, and that the combo box selection automatically updates the node.
- The dropdown list of the combo box[459] is created with a simple list of values.
- The **`Image Source Type`**[541] property of the Image control is set to `base64`.
- The **`Image Source`**[541] property of the Image control is set to the following XPath expression: `$XML1/images/element()[local-name() eq $PERSISTENT/Root/ComboBoxValue]`. This selects the child element of the `images` element that has a name that is equal to the content of the node `$PERSISTENT/ComboBoxValue`. In short, we select the Base64-encoded `image` element in the XML file that has a name that matches the content of the `$PERSISTENT/ComboBoxValue` node.
- So, when the end user selects an item in the combo box, that item's value is entered in the node `$PERSISTENT/ComboBoxValue`. The value in this node is then used to select the correct Base64 image element in the XML file. For example, if `png` is selected in the combo box, then `png` is entered as the value of the `$PERSISTENT/ComboBoxValue` node. The XPath expression of the **`Image Source`**[541] property then selects the `png` element of the XML file and displays its contents (the Base64-encoded PNG image) as the image.
- There is one important action still left. Every time a new value is selected in the combo box, we must specify that the image is reloaded[801]. Every time the image is reloaded, it reads the value in `$PERSISTENT/ComboBoxValue`, and retrieves the corresponding  image from the XML file.
- In the second table, the type of the image is obtained from the Base64-encoded text string by using the Altova XPath extension function `suggested-image-file-extension`[1718]. This function takes a string (the Base64 image) as its argument and retrieves the file-extension information from the string. If no file-extension information is available in the Base64 string, then the empty string is returned. The XPath expression used is:

```
for $k
in suggested-image-file-extension($XML1/images/element()[local-name() eq
$PERSISTENT/Root/ComboBoxValue])
return if ($k != '') then $k else "Data not available"
```

  The expression creates a variable (`$k`) that holds the file extension returned by the `suggested-image-`

file-extension[1718] function. If the variable is non-empty, then its content is displayed; otherwise, an appropriate message is displayed.

The next section, Exchangeable Image File Format (Exif)[1092], describes the remaining part of the design, which deals with Exif data.

## 12.2.4 Images Chosen by End User

The Let User Choose Image action[706] enables a solution to be designed in which the end user can choose an image to save to a data source. The image that the end user selects could be one that already exists in an image gallery (folder) or could be a photo that the user takes with the mobile device's camera application. In the second case, the Let User Choose Image action[706] automatically opens the camera application and saves the image that the user takes to the designated data source node. In both cases (gallery and camera), the image is added to the XML node as a Base64-encoded image.

A second action, Load/Save Image to File[707], saves an image in a data source node to an image file (with the appropriate image file extension).

The example file `UserSelectedImages.mtd` has a design that enables the end user to select an image from a gallery on the mobile device. This image is automatically saved to an XML node in the data source as a Base64-encoded string. The Base64-encoded image is then also automatically saved as an image file to a location selected by the designer (and defined in the design).

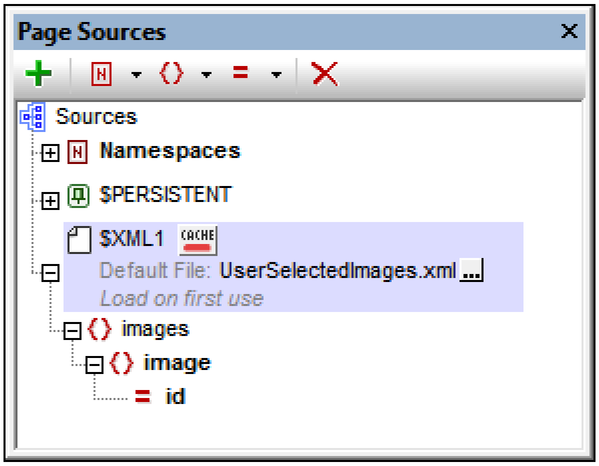
**Note:** If an image is displayed in the design, then, every time the image source is changed (for example, by a user selection), a Reload action[801] for the image is required in order to display the new image in the design.

### Example file: UserSelectedImages.mtd

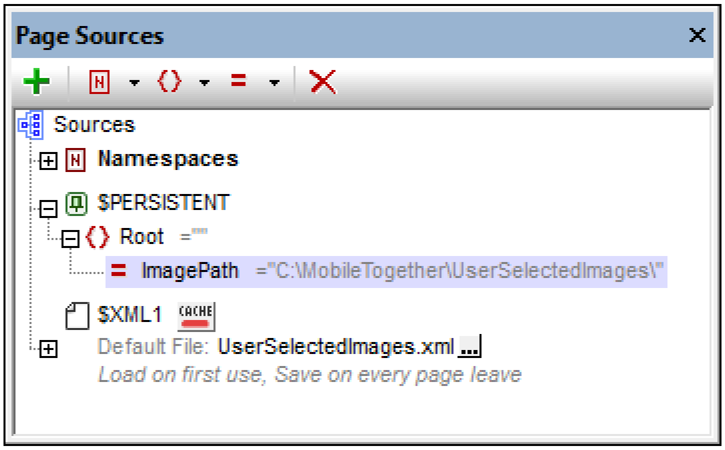
The design file `UserSelectedmages.mtd` is located in your *(*[72]*My) Documents*[72] MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\Images`**. You can open this file in MobileTogether Designer, run it in the simulator (**F5**), and look through the design definitions. You will also need to do the following:

- Create the folder **`C:\MobileTogether\UserSelectedImages`** since this is the folder that is defined in the design as the location where user selected images are saved. Alternatively, you can define some other save location for the Load/Save Image to File[707] action of the `OnImageClicked` event.
- On the MobileTogether Server settings page, set the *Server Side Solution's Working Directory* to be an ancestor directory of the design's default file, `UserSelectedImages.xml`. This ensures that the default file is updated when the Save to File[809] action of the `OnImageClicked` event is triggered. Since the Working Directory will be the base of all files referenced by the design, we suggest you set the Working Directory to **`C:\MobileTogether`**, and save the default file here. In this way, both the image folder and the default file folder are relative to the same base URI of **`C:\MobileTogether`**.

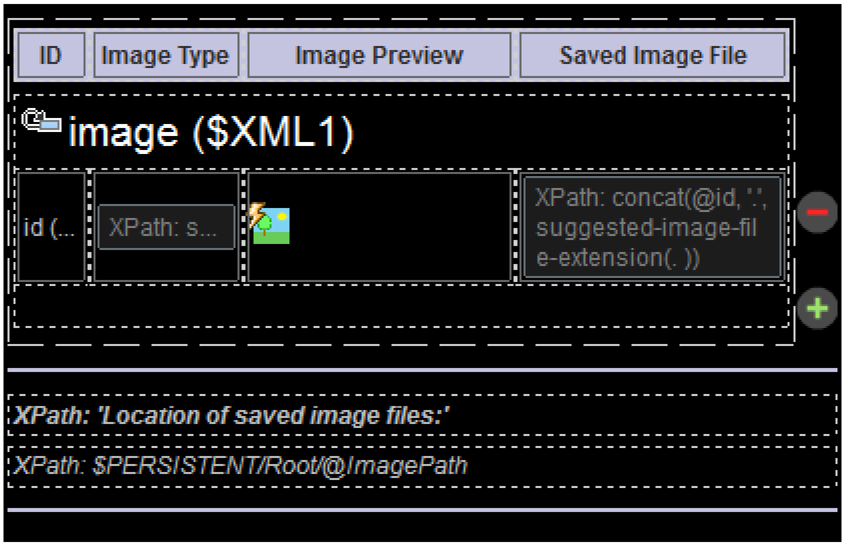
The design has a single data source, an XML file called `UserSelectedImages.xml` (also located in the `Tutorials\Images` folder). The structure of the XML document is shown in the screenshot below. The `images` element can have multiple **`image`** child elements. Each `image` element has an `id` attribute, and the `image` element's content will be the Base64-encoding of the image selected by the user. Every new image selected by a user is created in an automatically appended `image` element.


Page Sources
Sources
Namespaces
$PERSISTENT
$XML1
Default File: UserSelectedImages.xml
Load on first use
images
image
id

A node, **$PERSISTENT/Root/@ImagePath**, has been created to hold the path of the folder where images will be saved (*see screenshot below*). It has been set to a default value of **C:\MobileTogether\UserSelectedImages\**. You can modify this path directly in the Page Sources pane of the design if you wish to change the location of the folder where images are saved: double-click in the name and edit it.

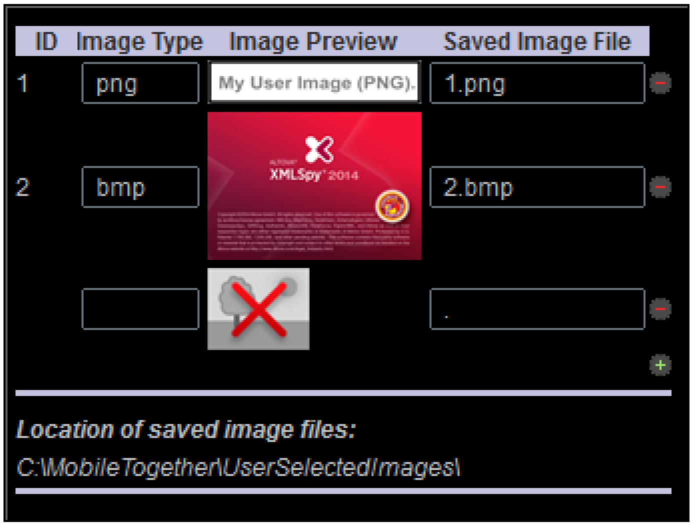

Page Sources
Sources
Namespaces
$PERSISTENT
Root =""
ImagePath ="C:\MobileTogether\UserSelectedImages\"
$XML1
Default File: UserSelectedImages.xml
Load on first use, Save on every page leave

The design (*screenshot below left*) consists of a label that displays the page title and a dynamic table[1069]. The dynamic table has a header row and a repeating row that is associated with the node **$XML1/images/image**. This means that the row repeats for every `image` element. Put another way, every `image` element is created in its own row. The screenshot below right shows the solution being run in a simulator. A description of the design is given below.



XPath: 'Location of saved image files:'

XPath: $PERSISTENT/Root/@ImagePath



| ID | Image Type | Image Preview | Saved Image File |
|---|---|---|---|
| 1 | png | My User Image (PNG). | 1.png |
| 2 | bmp | | 2.bmp |
| | | | . |

Location of saved image files:

C:\MobileTogether\UserSelectedImages\

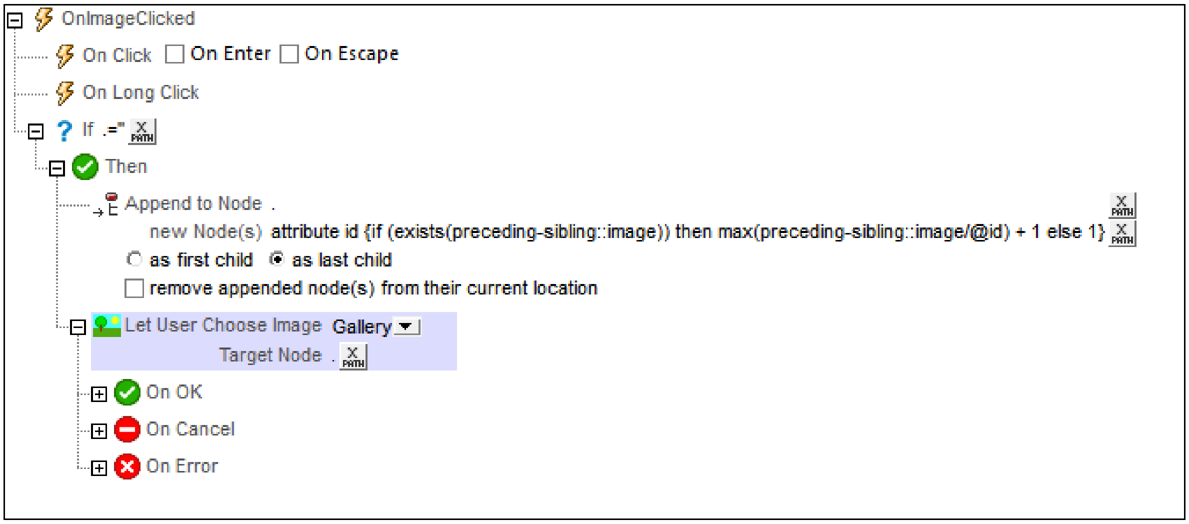
Note the following points about the design:

- The dynamic table has **Add/Remove** buttons (added via the table's context menu). This enables the user to add a new `image`-row and to delete any `image`-row.
- Each `image`-row has four columns: *ID*; *Image Type*; *Image Preview*, and (the name of the) *Saved Image File*.
- When a new `image`-row is added, a placeholder for the image is created and can be clicked (*see screenshot above right*).
- When the image placeholder is clicked, an `@id` attribute is added to the `image` element.
- The value of the `@id` attribute is calculated to always be one more than the largest existing image ID. This ensures the uniqueness of each ID value. If no preceding **`image`** element exists, then the added image will be the first **`image`** element and a value of `1` is assigned to the element's `@id` attribute. The XPath expression is defined with the `image` element as the context node: `attribute id {if (exists(preceding-sibling::image)) then max(preceding-sibling::image/@id) + 1 else 1}`.

- The *ID* column has a Label control[554] that is associated with the **`$XML1/images/image/@id`** node. This association (created by dropping the node onto the control) has the effect of displaying the value of the **`@id`** attribute of the current **`image`** element in the ID cell of the current row.
- The *Image Type* column has an Edit Field control[495] with an XPath expression that retrieves filetype information from Base64 text strings. The XPath expression submits the current node (the current **`image`** element) as the argument of the function **`suggested-image-file-extension`**[1718]. The function parses the Base64-encoded string for filetype information, and returns the file extension as a string.
- The *Image Preview* column contains the Image control[541]. The control has the `Image Source Type` property set to `base64` and the `Image Source` property set to the XPath expression `current()`. The current `image` element is the current node. The `Image Source Type` setting determines that the content of the image element will be read as a Base64 text (and not as a URL).
- The Image control[541] has a number of action defined for its `OnImageClicked` event. These are described in detail below.
- The fourth column, *Saved Image File*, gives the name of the image file that is saved to disk. It uses the Altova XPath extension function **`suggested-image-file-extension`**[1718] to provide the image file extension.

## Actions of the OnImageClicked event

- The `OnImageClicked` event of the Image control[541] has been assigned the actions shown in the screenshot below.



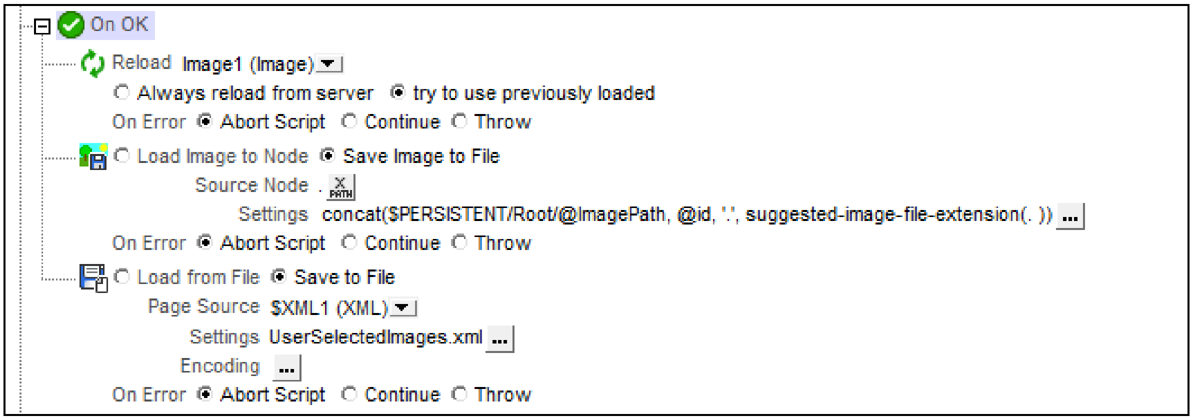
OnImageClicked
On Click □ On Enter □ On Escape
On Long Click
If .=''
Then
Append to Node .
new Node(s) attribute id {if (exists(preceding-sibling::image)) then max(preceding-sibling::image/@id) + 1 else 1}
○ as first child ● as last child
□ remove appended node(s) from their current location
Let User Choose Image Gallery
Target Node .
On OK
On Cancel
On Error

- The `If` condition specifies that if the current node (the `image` element) is empty, then, when the image is clicked, a new **`id`** attribute is created and appended[875] as a child to the currently empty `image` element. (The empty `image` element was added when the user added a table row (*see the Simulator screenshot further above*).) The `id` attribute is assigned a calculated value via the XPath expression: `if (exists(preceding-sibling::image)) then max(preceding-sibling::image/@id) + 1 else 1`. This expression returns a value that is always one more than the largest existing image ID, thus ensuring the uniqueness of each ID value. If no image is present, then the newly added image is assigned an ID value of `1`.
- The Let User Choose Image action[706] specifies that the image must be chosen from a folder on the mobile device (*Gallery*). This will allow the user to browse for an image when the image is clicked. The target node of the action is the location where the Base64-encoded image will be stored. In our example, the target node is the current node, which is the `Image` element.

- The Let User Choose Image action 706 has three conditions: `On OK`, `On Cancel`, and `On Error`, all of which are described separately below.

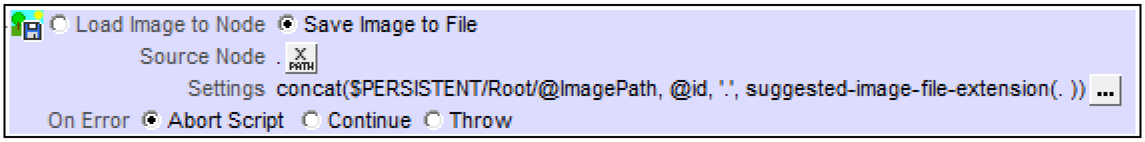
## On OK: Reload Image + Load/Save Image to File + Save to File

- The `On OK` condition defines three actions to carry out if the image is successfully imported to the designated data source node (*see screenshot below*): (i) a Reload action 801 for the image; (ii) a Load/Save Image to File action 707 that saves the image from the data source node to an image file; (iii) a Load/Save to File action 809 which saves the data in the source tree (on client/server) to the data source file.



On OK
Reload Image1 (Image)
○ Always reload from server ● try to use previously loaded
On Error ● Abort Script ○ Continue ○ Throw
○ Load Image to Node ● Save Image to File
Source Node .
Settings concat($PERSISTENT/Root/@ImagePath, @id, '.', suggested-image-file-extension(. )) ...
On Error ● Abort Script ○ Continue ○ Throw
○ Load from File ● Save to File
Page Source $XML1 (XML)
Settings UserSelectedImages.xml ...
Encoding ...
On Error ● Abort Script ○ Continue ○ Throw

- A Reload action 801 is set for the Image control 541. This will cause the image specified in the `Image Source` property of the Image control 541 to be reloaded. Since the value of the `Image Source` property is set to the current node, and since the current node is the `Image` element which is the target node of the user-selected image, the image preview cell for the current row will be reloaded with the user's image.
- The Load/Save Image to File action 707 (*screenshot below*) saves the image from the data source node to an image file. The *Source Node* has been set to the current node (which is the `image` element). The binary image file will be generated from the Base64 data in this node.

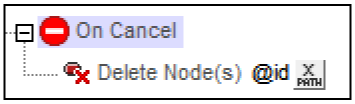


○ Load Image to Node ● Save Image to File
Source Node .
Settings concat($PERSISTENT/Root/@ImagePath, @id, '.', suggested-image-file-extension(. )) ...
On Error ● Abort Script ○ Continue ○ Throw

- The *Settings* option specifies the location where the binary image file will be saved. The XPath expression generates the location where the image is to be saved, and the filename of the image. It specifies the node in the **`$PERSISTENT`** tree that holds the path to the image folder; the **`@id`** attribute provides the filename, and the Altova XPath extension function **`suggested-image-file-extension`** 1718 provides the file extension.
- The Load/Save to File action 809 saves the data in the data source tree on the server to the specified data source file.

## On Cancel: Delete Node

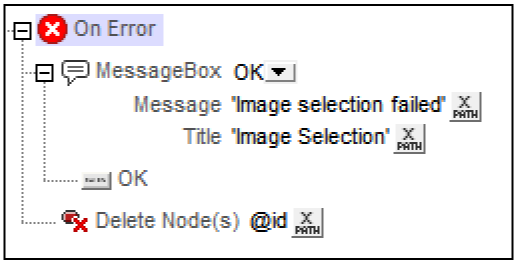
If the user decides to cancel the image-selection process, the `@id` node is deleted with the Delete Node(s) action[879]. Remember: The `@id` node was created when the image-selection process was started (by clicking the image placeholder; *see the "Example File" section above*).



On Cancel
Delete Node(s) @id

## On Error: MessageBox + Delete Node

If there is an error while importing the image as Base64 data to the designated XML node, the actions defined for the `On Error` condition are executed. An error message is displayed and the `@id` node is deleted. The `@id` node was created when the image-selection process was started (*see the "Example File" section above*).



On Error
MessageBox OK
Message 'Image selection failed'
Title 'Image Selection'
OK
Delete Node(s) @id

## 12.2.5 Transforming Images

A Base64-encoded image[1098] can be transformed (resized, rotated, and modified for quality/filesize) with the Altova XPath extension function **`mt-transform-image`**[1718]:

```
mt-transform-image(Base64Image as Base64BinaryString, Size as item()+, Rotation as
xs:integer, Quality as xs:integer) as Base64BinaryString
```

The function takes a Base64-encoded image as its first argument and returns a transformed Base64-encoded image. The second, third, and fourth arguments are the image parameters that are transformed: size, rotation, and quality. For a detailed description of the function and usage examples, see the section XPath/XQuery Functions: Image-Related[1718].

Note the following points:

- The input image for the transformation is a Base64-encoded image—not an image file.
- Any Exif data in the Base64-encoding will be lost in the transformed image.
- If the transformation is done on the client, there could be memory issues on the client. *See note below.*

> *Transformation on client or server*
> The function **`mt-transform-image`**[1718] will be executed on the client if not explicitly specified otherwise. This could create memory problems on some clients. When the transformation is started, the image is

unpacked from the format of its Base64 encoding to a BMP format, which could be very large. After the transformation is completed, the transformed file is stored back to the original format. The large BMP format could create memory problems on some clients, and you should be aware of this.

To avoid the possibility of memory problems on the client, explicitly specify that the transformation must be carried out on the server. Do this with the Execute On action[914], specifying that the child actions be performed on the server. All the child actions of this Execute On action[914] will then be carried out on the server. You can use an action such as Update Node[886] to update a node with the result of a transformation. The target node will be updated with the transformed image. MobileTogether automatically transfers the results to the client when action handling is complete or when the workflow switches back to the client.

## 12.2.6 Images in Databases

Images in databases can be stored in Base64 format, which is a text format into which binary data can be encoded.

# 12.3 Audio, Video

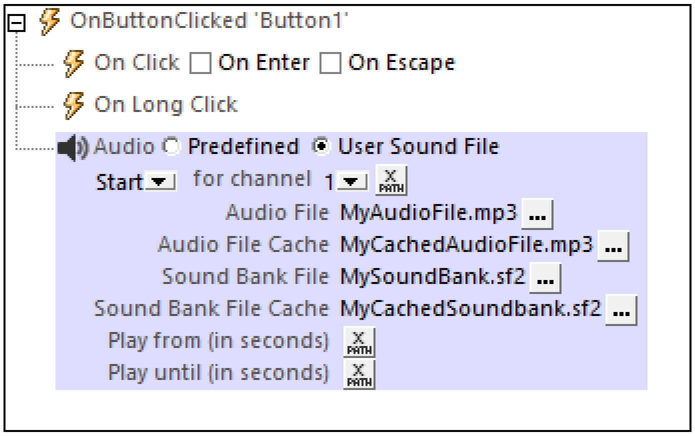
This section provides an overview of MobileTogether's audio and video features. It is organized into the following sections:

- Audio Playback 1108
- Audio Recording 1110
- Video Playback 1112
- Audio/Video Formats 1115

## 12.3.1 Audio Playback

MobileTogether's Audio playback feature enables predefined audio sounds (available on clients) or audio files located on the mobile device or at a remote location to be played back. Audio can be played back on five channels (numbered 1 to 5), and each Audio action is defined for one specific channel.

Audio playback is controlled via the Audio (Playback) action 720. Each Audio action defines one of the following: (i) start playback of a predefined sound or specified file on a specified channel (*see screenshot below*), (ii) pause playback on a specific channel, (iii) resume playback on a specific channel, (iv) stop playback on a specific channel, and (v) jump to a position in the audio file playing on a specific channel. See the description of the Audio action 720 for details. Typically, each Audio action will be assigned to a control event, such as a button click. When the event occurs, the Audio action is triggered.



OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

Audio ○ Predefined ● User Sound File

Start for channel 1

Audio File MyAudioFile.mp3

Audio File Cache MyCachedAudioFile.mp3

Sound Bank File MySoundBank.sf2

Sound Bank File Cache MyCachedSoundbank.sf2

Play from (in seconds)

Play until (in seconds)

The Audio playback feature works as follows:

- When the Audio Start action is triggered, the selected predefined sound or the audio file named in the action will be played on the channel specified.
- You can choose from among the following predefined sounds (which are available on client devices): `ClickOffOn`, `ClickOnOff`, `Ding`, `DingDong`, `ErrorDeepBuzz`, `ErrorWhoops`, `Goodbye`, `KeyClickTick`, `KeyClickTock`, `MessageBounce`, `MessageXylophone`, `WhooshDeep`, `WhooshExhale`, `WhooshLong`, `WhooshQuick`, `WhooshQuicker`.
- If you choose to play an audio file, then the audio file, if located on the client device, will play directly. If the file is located on a remote server, then the file will be downloaded to the client device. If a local

cache file is specified in the settings, then the downloaded data will be saved to this local file. If the specified cache file already exists, then the cache file will be played and no download takes place.

- MIDI file playback is supported on all client devices except Web browsers. On iOS devices, however, MIDI file playback requires a sound bank file. The location of this file must be entered in the Start action (*see screenshot above*).
- In the Start action, you can specify whether the entire audio file should be played back, or only a segment of it. The segment is defined in terms of start and end times (*see screenshot above*).
- Each Start action is assigned to a channel, numbered 1 to 5. You can therefore run up to five audio streams simultaneously. Each Audio action is defined for a specific channel, and the settings of the action will apply to that channel.
- The Pause, Resume, and Stop actions are simple actions, each of which would typically be defined on a control such as a button. Each of these actions is defined for a particular channel. They carry out the respective action for the audio file playing on that channel.
- The Seek To action applies to the audio file playing on the specified channel, and jumps to the specified position in that file.

**Note:** If an audio stream is playing when a solution is suspended 915, then playback is paused. Playback continues when the solution is resumed.

**Note:** Multi-channel audio/video playback is not supported on Windows Phone. Only one audio **or** video file can be played at a time: this is the file that was started last.

**Note:** Audio and video files **cannot** be deployed to MobileTogether Server via the MobileTogether Designer project's Deploy to Server mechanism 291. You can, however, copy audio/video files manually to the server, although you cannot stream them from there via a URL. If you wish to stream audio/video files that are located on your MobileTogether Server, then do the following: (i) use the Load Binary 815 action to load the binary audio/video data to a page source node; (ii) use the Save Binary 815 action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 1108 to play the file that is now saved on the client device. Alternatively, you can save audio/video files to a web server—instead of saving to MobileTogether Server—and use a URL to stream the audio/video file from the web server.

## Audio playback events

Audio playback events are defined for the entire project. There are three predefined audio playback events (*listed below*). A set of any of MobileTogether's available actions can be defined for each of these events. To access the dialog in which these event actions are defined, click the **Additional Dialog** button of the `Audio Actions` 296 project property. Since each of these events apply across the entire project, each event could be triggered by the audio on any channel. The `$MT_AudioChannel` 1304 dynamic variable contains the number of the channel that triggered the event. So, for example, if the user starts playback of an audio file that runs on Channel 2, you can use the `$MT_AudioChannel` 1304 variable in an XPath expression of an **`OnAudioStarted`** event action. The action could, for example, display database information about the audio file running on Channel 2.

- **`OnAudioStarted`**: Before this event occurs (that is, before the audio file starts playing), **details of the audio file are not available**, and the functions to get audio duration and current position (*see below*) should not be called; at this time, only the **`mt-audio-is-playing`** 1262 function will return valid information. The **`OnAudioStarted`** event can be used, for example, to log details of audio playback to an XML tree node (say, via the Update Node action 886).
- **`OnAudioError`**: Possible errors could be: File not Found, a file format error, or download/playback interruption. Information about the error can be retrieved with the MobileTogether XPath extension function `mt-external-error` 1262. If actions are defined for the event, these actions are executed. Otherwise, the error is shown in a message box.

- `OnAudioCompleted`: Audio playback is considered to be completed when the file or specified segment plays to the end (without a Stop action being executed). The actions defined for this event are **not** performed when the audio is suspended (with the project property *On Switch to Other Solution*[296]) or paused.

### Audio-playback-related MobileTogether XPath extension functions

The following audio-playback-related MobileTogether XPath extension functions[1262] are available:

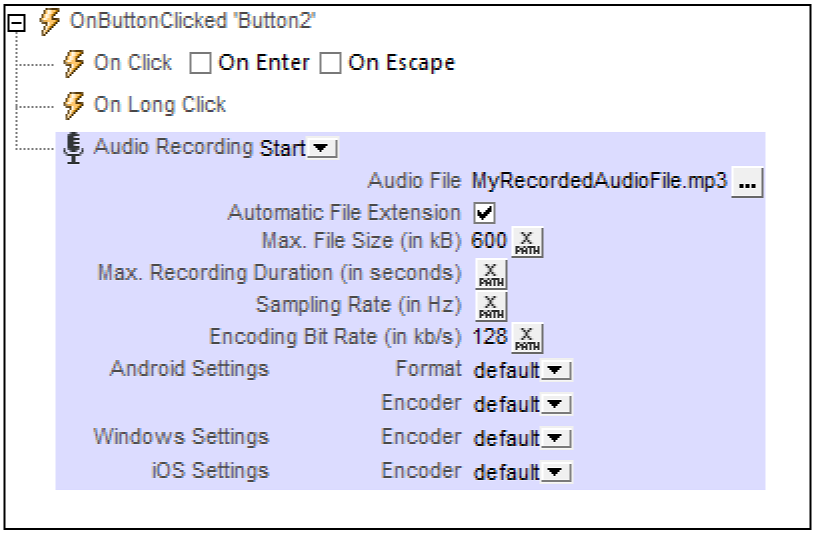
- **`mt-audio-get-current-position(`** ***`ChannelNumber`*** as `xs:integer` **`)`** as `xs:decimal`
- **`mt-audio-get-duration(`** ***`ChannelNumber`*** as `xs:integer` **`)`** as `xs:decimal`
- **`mt-audio-is-playing(`** ***`ChannelNumber`*** as `xs:integer` **`)`** as `xs:boolean`

You can use these functions in XPath expressions anywhere in the design, for example, to display to the user the current position of audio playback in seconds. Note that before audio playback starts, details about the audio file are not available. As a result, information such as the duration and current position will not be known. The corresponding functions should therefore only be used after playback starts.

## 12.3.2 Audio Recording

MobileTogether's Audio Recording feature enables audio to be recorded via the client device's audio recording app and be saved to a file on the client device.

Audio recording is started via the Audio Recording (Start) action[724] (*see screenshot below*) and stopped via the Audio Recording (Stop) action[724]. See the description of the Audio Recording action[724] for details. Typically, each Audio Recording action will be assigned to a control event, such as a button click. When the event occurs, the Audio Recording action is triggered. For example, one button can be used to start the recording and another can be used to stop the recording. Recording is also automatically stopped when the end user leaves the page on which the audio recording was started or when the solution is suspended[915].



OnButtonClicked "Button2"

On Click □ On Enter □ On Escape

On Long Click

Audio Recording Start

Audio File: MyRecordedAudioFile.mp3

Automatic File Extension: ☑

Max. File Size (in kB): 600

Max. Recording Duration (in seconds):

Sampling Rate (in Hz):

Encoding Bit Rate (in kb/s): 128

Android Settings — Format: default; Encoder: default

Windows Settings — Encoder: default

iOS Settings — Encoder: default

The Audio Recording feature works as follows:

- When the Audio Recording Start action is triggered, the device's recording app is started and audio is recorded to the file named in the action (*see screenshot above*). This file must be on the client device.
- You can specify which encoder (codec) is used for each operating system. If you leave these settings blank, then the default codec on each device will be used. For Android systems, you can also specify the file format of the recorded file.
- You can specify the file size and recording duration of files that are recorded. If one of these limits is reached, then recording is stopped and is considered to have been completed.
- You can also specify the sampling rate and encoding bitrate of recordings. If you leave these settings blank, then the default settings of the recording codec will be used. If you wish to specify your own settings, be sure to consult the related audio encoding standard or the encoder specifications.
- The Audio Recording Stop action stops any audio recording that was started on that page.

**Note:** If audio is being recorded when the end user leaves the page or when a solution is suspended 915, then recording is stopped. If a second recording action is started while one is in progress, then the first recording action is stopped. The first recording action is considered to be interrupted, that is, unfinished.

**Note:** Audio should not be recorded at the same time as audio/video is being played back as this could result in problems with the playback state, particularly on iOS devices.

### Audio recording events

Audio Recording events are defined per page. Two events are available: `OnAudioRecordingError` and `OnAudioRecordingFinished`. The actions that are defined for these events **apply to all Audio Recordings on the page**. You can access these events by either (i) clicking the **Additional Dialog** button of the Audio Recording Actions property 384, or (ii) right-clicking in the design and selecting **Page Audio Recording Actions**. For each event, you can define the actions to perform by dragging and dropping actions from the left-hand Actions pane into the event's tab.

- **`OnAudioRecordingError`**: Possible errors could be: File not Found, a file format error, or a recording interruption. Information about the error can be retrieved with the MobileTogether XPath extension function `mt-external-error` 1262. If actions are defined for the event, these actions are executed. Otherwise, the error is shown in a message box.
- **`OnAudioRecordingFinished`**: Audio recording is considered to be finished when the maximum file size (**`Max File Size`**) or maximum duration (**`Max Recording Duration`**) has been reached (*see screenshot above*).

### Audio-recording-related MobileTogether XPath extension functions

The following audio-recording-related MobileTogether XPath extension function 1262 is available:

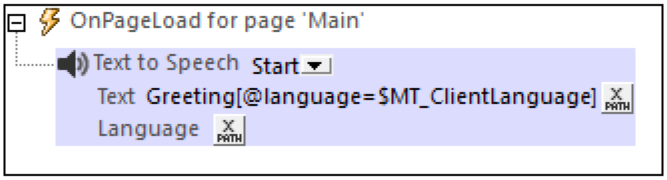
- **`mt-audio-is-recording()`** as `xs:boolean`

You can use this function in XPath expressions, for example, to specify processing that is conditional on whether audio is currently recording or not.

## 12.3.3 Text to Speech

The Text to Speech feature converts a text string to audio and plays it back. The text string to play back can be specified directly in the Start settings of the Text to Speech action 727 (*see screenshot below*) or via an

XPath expression. The action's *Language* setting is set by default to the language setting of the mobile device. It can be used to override the device's language setting. For more details, see Text to Speech action 727.


OnPageLoad for page 'Main'
Text to Speech Start
Text Greeting[@language=$MT_ClientLanguage]
Language

The *Stop* mode of the action, when executed stops any currently playing Text to Speech playback.

**Note:** Text to Speech playback is available on mobile devices only and cannot be simulated on MobileTogether Designer.

### Text to Speech events

The Text to Speech actions of the project properties 296 enable actions to be set on the following events: `OnTextToSpeechStarted`, `OnTextToSpeechError`, `OnTextToSpeechCompleted`. These events enable further action to be taken at those points in time when these events are triggered.

### XPath functions related to Text to Speech

The following text-to-speech-related MobileTogether XPath extension functions 1262 are available:

- **`mt-text-to-speech-is-language-available(`*`language`*`)`** as `xs:boolean`
- **`mt-text-to-speech-is-speaking()`** as `xs:boolean`

You can use these functions in XPath expressions to test whether the conditions they define are fulfilled. The action to be taken by the design can thus be made conditional on these environment variables.

## 12.3.4 Video Playback

MobileTogether's Video playback feature enables: (i) remote video files to be directly streamed to the client device, and (ii) locally saved video files to be played. Video playback is defined in two steps:

1. A Video control 728 is used to set up the viewing window on the page and to specify the URL of the video file to download. See the description of the Video control 728 for details.
2. Video actions 728 specify the playback action to perform : *Start Video, Pause, Resume, Stop, Seek (Jump) To*.

### Setting up the video window and video file

You can insert multiple Video controls 728 on a page. Each Video control 728 is identified by a name, and is assigned a video source via a URL. The name of a Video control 728 will be used in the Video action 728 to indicate on which Video control 728 the action is to be performed.

The following Video control 728 properties are used to define key attributes of the control:

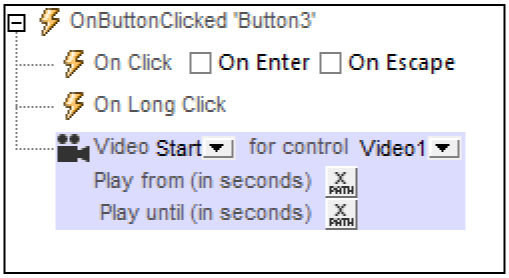
- **`Play on Load`**: Specifies whether the video is played as soon as the page has loaded. If playback is to be started at a later time, use the Video Start action 728 (for example, on a Button 417).
- **`Video Source`**: Specifies the remote or local video file to play.
- **`Cached Video Source`**: The URL on the client device where the cached video file is saved. If no cache file exists at this location, one is created when the video source file is downloaded for playback. If a cache file does exist, then the cache file is played, and no new download takes place.
- **`Show Controls`**: Determines whether video playback buttons are displayed within the control. This would enable the end user to control playback actions, for example to start, pause, resume, and stop playback. If this property is set to **`false`**, then playback actions should be provided via Video actions 728. Note that Video control buttons are not supported on Windows Phone.
- **`Initial Width`**: Sets the initial width of the control. When the video starts, the control resizes to the actual width if the control's **`Control Width`** property has been set to `wrap_content`. If the **`Control Width`** property is set to `fill_parent`, then the complete width (of the parent) is used and only the height is adjusted.
- **`Initial Height`**: Sets the initial height of the control. When the video starts, the control resizes to the actual height.

**Note:** Multi-channel audio/video playback is not supported on Windows Phone. Only one audio **or** video file can be played at a time: this is the file that was started last.

**Note:** Audio and video files **cannot** be deployed to MobileTogether Server via the MobileTogether Designer project's Deploy to Server mechanism 291. You can, however, copy audio/video files manually to the server, although you cannot stream them from there via a URL. If you wish to stream audio/video files that are located on your MobileTogether Server, then do the following: (i) use the Load Binary 815 action to load the binary audio/video data to a page source node; (ii) use the Save Binary 815 action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 1108 to play the file that is now saved on the client device. Alternatively, you can save audio/video files to a web server—instead of saving to MobileTogether Server—and use a URL to stream the audio/video file from the web server.

## Video playback actions

Each Video action 728 (*screenshot below*): (i) identifies the Video control 728 to which it applies (via the video control's name, and (ii) specifies the action to perform on the video file that is associated with the control. These actions are: *Start, Pause, Resume, Stop, Seek (Jump) To*. The Video action 728 also allows you to specify that a specific file segment should be played instead of the entire file. For details, see the description of the Video action 728.



OnButtonClicked 'Button3'
On Click ☐ On Enter ☐ On Escape
On Long Click
Video Start for control Video1
Play from (in seconds)
Play until (in seconds)

**Note:** If a video stream is playing when a solution is suspended 915, then playback is paused. Playback continues when the solution is resumed.

### Video playback events

Video playback events are defined on each Video control[728] and apply to that Video control[728]. You can access these events either via the control's context menu (right-click to open) or the video control's `Control Action` property. For each event, you can define the actions to perform by dragging and dropping actions from the left-hand Actions pane into the event's tab.

- **`OnVideoStarted`**: Before this event occurs (that is, before the video starts playing), **details of the video file are not available**, and the functions to get video height, width, duration, and current position (*see below*) should not be called; at this time, only the **`mt-video-is-playing`** function will return valid information. This event can be used, for example, to log details of video playback (say, via the Update Node action[886]) to an XML tree node.
- **`OnVideoError`**: Possible errors could be: File not Found, a file format error, or download/playback interruption. Information about the error can be retrieved with the MobileTogether XPath extension function `mt-external-error`[1262]. If actions are defined for the event, these actions are executed. Otherwise, the error is shown in a message box.
- **`OnVideoCompleted`**: Video playback is considered to be completed when the file or specified segment plays to the end (without a Stop action being executed). The actions defined for this event are **not** performed when the video is suspended (with the project property *On Switch to Other Solution*[296]*)* or paused.

### Video-related MobileTogether XPath extension functions

The following video-related MobileTogether XPath extension functions[1262] are available:

- **`mt-video-get-current-position(`** ***`VideoControlName`*** as `xs:string` **`)`** as `xs:decimal`
- **`mt-video-get-duration(`** ***`VideoControlName`*** as `xs:string` **`)`** as `xs:decimal`
- **`mt-video-height(`** ***`VideoControlName`*** as `xs:string` **`)`** as `xs:integer`
- **`mt-video-width(`** ***`VideoControlName`*** as `xs:string` **`)`** as `xs:integer`
- **`mt-video-is-playing(`** ***`VideoControlName`*** as `xs:string` **`)`** as `xs:boolean`

You can use these functions in XPath expressions, for example, to specify processing that is conditional on the height/width of the video. Note that before video playback starts, details about the video file are not available. As a result, information such as the height, width, duration and current position will not be known. The corresponding functions should therefore only be used after playback starts.

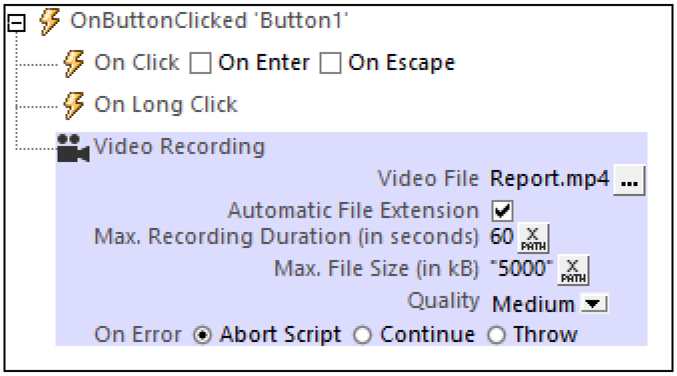
## 12.3.5 Video Recording

MobileTogether's video recording feature enables: (i) the mobile device's recording app to be started from the MobileTogether solution when an event in the solution is triggered, and (ii) the file of the video recording to be automatically saved at a designated location with a designated file name when the recording is stopped.

The feature enables the user of the mobile device to directly record video with the device's video recording app (after this has been started by the video recording action). When the user stops the recording, or when the maximum recording duration or recording file size has been reached, the video file is saved to the designated location.

## Setting up a video recording

A video recording is defined in a Video Recording[729] action *(screenshot below)*. This action can be set within a sequence of actions to perform when a particular event (page event[389] or control event[665]) is triggered.



⊟ OnButtonClicked 'Button1'

On Click □ On Enter □ On Escape

On Long Click

Video Recording

Video File Report.mp4 ...

Automatic File Extension ☑

Max. Recording Duration (in seconds) 60

Max. File Size (in kB) "5000"

Quality Medium

On Error ● Abort Script ○ Continue ○ Throw

In the Video Recording[729] action *(screenshot above),* you can define the following properties:

- The name and location of the video file to save when the recording has been ended.
- The maximum recording duration or file size. This can be used to restrict the duration or size of the recording.
- The picture quality of the video recording.

For details of these properties, see the description of the Video Recording[729] action.

## Useful MobileTogether XPath extension functions

The following MobileTogether XPath extension functions can be useful (for example, to find the location of the recorded file if this has not been specified in the properties of the Video Recording[729] action, or to find the file extension of the video file) :

- `mt-last-file-path`[1262] gets the file path and file name of the recording. You can use this function not only to determine the location of the recorded-video file, but also to submit this file path as the argument of the next two functions.
- `mt-extract-file-name`[1262] gets the file name from the submitted file path.
- `mt-extract-file-extension`[1262] gets the file extension from the submitted file path.

## Note about simulations

For simulations, you will be able to select a video recording from disk to use instead of the recorded video file, and action handling will continue.

# 12.3.6 Audio/Video Formats

Given below are links to Internet pages that contain information about the audio and video formats supported by the various client devices that run MobileTogether solutions. We have selected web pages that we find the most useful. However, if the information you require is not available on the pages listed below, please search for alternative sources.

**Note:** MIDI file playback is supported on all client devices except Web browsers. On iOS devices, however, MIDI file playback requires a sound bank file.

## Android

Android Developer Website: Supported Media Formats

*Audio recording formats*

In general, the supported formats/codecs for audio recording will depend on the device and the OS version. Because of the large number of device-OS combinations that are available it is best not to select a specific format or encoder, and, instead, to use the default format/codec of the respective devices. Note the following restrictions:

- For recording to `AMR-NB` and `AMR-WB` audio formats, use only the AMR-NB and AMR-WB codecs, respectively.
- The `AAC` audio format can be recorded with the AAC-Low Complexity, High Efficiency-AAC, and Enhanced Low Delay-AAC codecs, but it is possible that not all of these codecs will be available or work on a given device.

## iOS

iOS supports many industry-standard video formats and compression standards, including the following:

- H.264 video, up to 1.5 Mbps, 640 by 480 pixels, 30 frames per second, Low-Complexity version of the H.264 Baseline Profile with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in `.m4v`, `.mp4`, and `.mov` file formats
- H.264 video, up to 768 Kbps, 320 by 240 pixels, 30 frames per second, Baseline Profile up to Level 1.3 with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in `.m4v`, `.mp4`, and `.mov` file formats
- MPEG-4 video, up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in `.m4v`, `.mp4`, and `.mov` file formats
- Numerous audio formats, including the ones listed in Audio Technologies

Apple Developer Website: Audio Information
Apple Developer Website: Video Information

## Windows

Audio/Video codecs and formats for Windows

*Audio recording formats*

WMA and MP3 files are not supported as recording formats on Windows Phone.

## Web browser

Basic overview of playing audio in HTML
Basic overview of playing video in HTML
Support of audio/video formats in various browsers

The best supported formats currently are:

- *Audio:* **`mp3`**, **`aac`**
- *Video:* **`H.264 (mp4)`**

# 12.4 NFC

**Near Field Communication (NFC)** is a set of wireless technologies which allows for the transfer of data across short distances—typically 4cm (1.5 inch) or less—between two NFC-enabled devices. The technology is most commonly seen in use when payments are made by touching an NFC-enabled credit/debit card to an NFC-enabled payment terminal. NFC allows the transfer of small payloads of data, usually text or numbers. But NFC can also be used to transfer other kinds of data (such as images and files) between two NFC-enabled devices.

For more information about NFC, see nearfieldcommunication.org, Wikipedia, and the NFC Forum.

## NFC-enabled devices

NFC-enabled devices may be active or passive. A passive device—for example an NFC tag in a credit card—contains information that other (active) NFC devices, such as smartphones, can read. Active devices can read information and send information. A smartphone is an active NFC device. It can read information from passive NFC devices as well as exchange information with other NFC-enabled devices.

If security of communication is important, NFC can establish a secure connection and use encryption.

## NDEF technology

NFC data is sent and received in the form of **NFC Data Exchange Format (NDEF)** messages. In the NDEF format, each communication is an **NFC tag**. Each NFC tag contains an **NDEF message**, and each NDEF message contains one or more **NDEF records**. When an active NFC device is unlocked, it will automatically search for NFC tags in its vicinity. Depending on the intent of any discovered NFC tag, the device will determine how to best handle the NFC tag. It is important that the device does **not** ask the user what action to take. This is because any user input will cause the device to be moved away from the NFC tag, thus breaking the connection. For more information, see the Android Developer Guide.

One important point to keep in mind while designing for NFC in MobileTogether is that the **payload** of the NFC message (that is, the content of the message) is stored and transferred in hexBinary format. The lexical space of the hexBinary format is a simple coding (of data points) as hexadecimal values. For example: The string `hi` when converted to hexBinary format is `6869` (since the hex representation of `h` is `68` and the hex value of `i` is `69`).

## NFC tags

The term NFC tag is used to refer to two different concepts:

- a **piece of data** that is transferred using the NDEF technology (see *NDEF technology* immediately above)
- a passive NFC **device** that contains NFC data

The second kind of NFC tag listed above is a hardware object that contains a microchip. In its simplest form, this kind of NFC tag resembles a postage label. The most significant properties of this kind of NFC tag are: (i) it contains data that can be read; (ii) the data it contains can be overwritten multiple times till the NFC tag is locked; (iii) once it is locked, the NFC tag cannot be overwritten any more.

For more information, see this article: How NFC Tags and Readers Work.

## NFC availability on Android, Windows, and iOS

- *Android*: To check the availability of NFC on an Android device and enable it, go to: **Settings | Connected Devices | Connection preferences | NFC**.
- *Windows*: To check the availability of NFC on a Windows device and enable it, go to: **Settings | Tap+Send**.
- *iOS*: NFC is used primarily with Apple Pay.

**Note:** If both an Android device and a Windows device are NFC-enabled, then the Windows device can send data[1120] to the Android device via NFC.

## MobileTogether and NFC

MobileTogether solutions support NFC in the following ways:

- NFC tags can be read and data from them can be processed further (Android and Windows devices).
- Messages can be pushed from a Windows device to an NFC-enabled receiving device.

**Note:** NFC support in MobileTogether is not available on iOS devices.

## Note about Android Beam™

Android Beam™ is an app that is available on Android devices since Android 4.0. It enables the sharing of data between two Beam-enabled devices. Android Beam has been deprecated since Android 10.

Android Beam was supported in MobileTogether from versions 3.2 to 8.1. However, starting with MobileTogether 9.0, support for Android Beam has been discontinued and any push functionality[1120] that used Android Beam must be removed.

## In this section

# 12.4.1 Discovering and Reading NFC Tags

When an NFC Start/Stop action[746] is added to the design, a page source named `$MT_NFC` is automatically added to the design. It is this page source that will, at run time, hold the NFC data that is discovered in an NFC tag.

At run time, if NFC has been started via a solution's NFC Start action[746], then NFC tag discovery starts automatically. If a tag is discovered, then the NFC message in it will be automatically received and the information in the message will be stored in the `$MT_NFC` tree. The complete structure of the tree is given below. Note that an NDEF message can contain multiple NDEF records, and the NDEF records can be recursive. If the NFC tag information that is received does not contain information to fill all the attributes of the **`NDEFMessage`** or **`NDEFRecord`** elements, then these attributes will not be created in the `$MT_NFC` tree.

⊟ *Complete structure of $MT_NFC tree*

```
<Root>
   <Tag Id=""/>
   <NdefMessage
      CanMakeReadOnly=""
      IsWriteable=""
      MaxSize=""
      Type="">
      <NdefRecord
         Id=""
         TypeNameField=""
         RecordTypeDefinition=""
         Type=""
         Text=""
         Language=""
         URI=""
         Payload=""
         MimeType=""
         ExternalDomain=""
         ExternalPackageName="">
         <NdefRecord />
      </NdefRecord>
      <NdefRecord />
         ...
      <NdefRecord />
   </NdefMessage>
</Root>
```

The information in the `$MT_NFC` tree can be processed further and displayed in the same way as other page source data. For example, the `$MT_NFC/Root/Tag/NdefMessage/NdefRecord/@Text` node can be linked to a label in order to display the message text in the label.

**Note:** Additional actions to take when an NFC tag is discovered can be specified via the tab of the **`OnNfcTagDiscovered`**[1121] event.

**Note:** Information from an NFC tag will overwrite any information that might already exist in the `$MT_NFC` tree. So each subsequently discovered tag will replace the information of the previous tag in the `$MT_NFC` tree.

## 12.4.2 Pushing Data to Other Devices

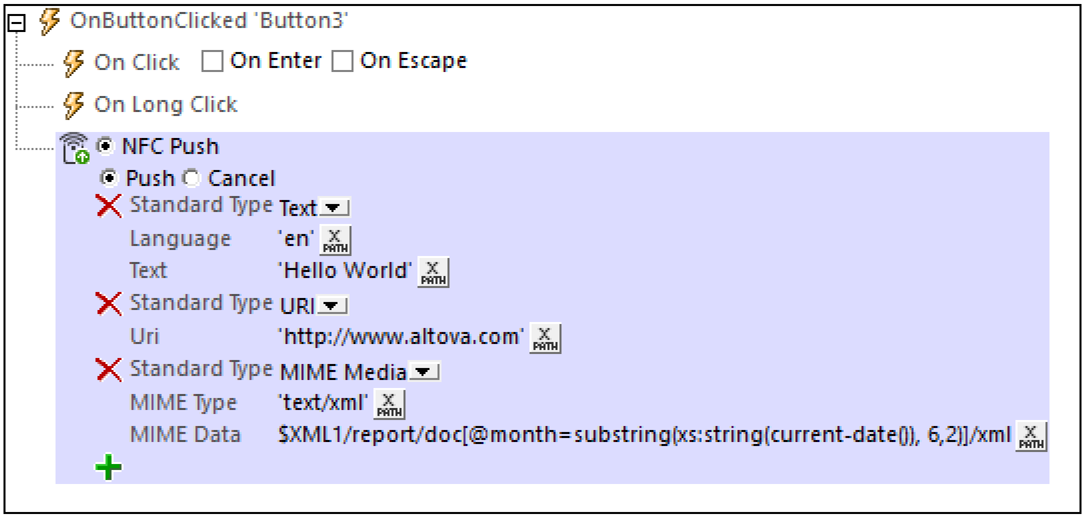
A MobileTogether solution can be used to transmit data from the Windows device running the solution to another NFC-enabled device. The steps of the procedure are as follows:

1. On the sending device, start the solution.
2. Trigger the NFC Start[746] action (for example, by tapping a button).

3. Place the back of the sending device against the back of the receiving device.
4. Trigger the NFC Push[747] action on the sending device to start transmission (for example, by tapping a button). Once an NFC Push action has been started, the NFC message or file will be transmitted to any receiving device that is placed within NFC range of the sending device. This continuous sending is stopped (i) when NFC is stopped[746] or (ii) when an NFC Push[747] action is canceled (which is done by adding a new NFC Push action with the *Cancel* option selected; *see the action's screenshot below*).

## How it works

NFC data-transmission is defined in the NFC Push[747] action (*see screenshot below*). In the action's *Push* option (*see screenshot*), specify the message type (for example, *Text* or *URI*) and message to send. For details, see the description of the NFC Push action[747].



OnButtonClicked 'Button3'
On Click □ On Enter □ On Escape
On Long Click
● NFC Push
● Push ○ Cancel
Standard Type Text
Language 'en'
Text 'Hello World'
Standard Type URI
Uri 'http://www.altova.com'
Standard Type MIME Media
MIME Type 'text/xml'
MIME Data $XML1/report/doc[@month=substring(xs:string(current-date()), 6,2)]/xml

**Note:** NFC data-pushing is supported on Windows devices, but not on Android or iOS devices.

## 12.4.3 NFC-Related Events

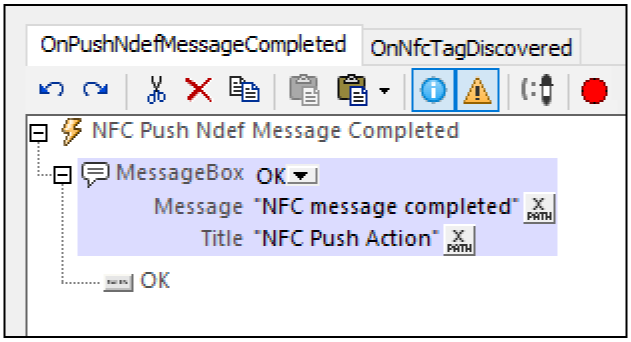
You can specify what actions to perform when two NFC-related events (*see screenshots below*) are triggered.

- `OnPushNdefMessageCompleted` specifies what action/s to carry out when the transmission of NFC data (via NFC Push[747]) has been completed.
- `OnNfcTagDiscovered` specifies what (additional) action/s to carry out when an NFC tag is discovered[1119].

To access these actions, go to Styles & Properties Pane[274] | Project Properties[296] | NFC Actions property, and click the property's **Additional Dialog** button.

### OnPushNdefMessageCompleted

This event is triggered when a message or file has been successfully transmitted[1120] and can be used to specify subsequent action to take. For example, the solution's user can be informed about the completion of transmission, as shown in the screenshot below.



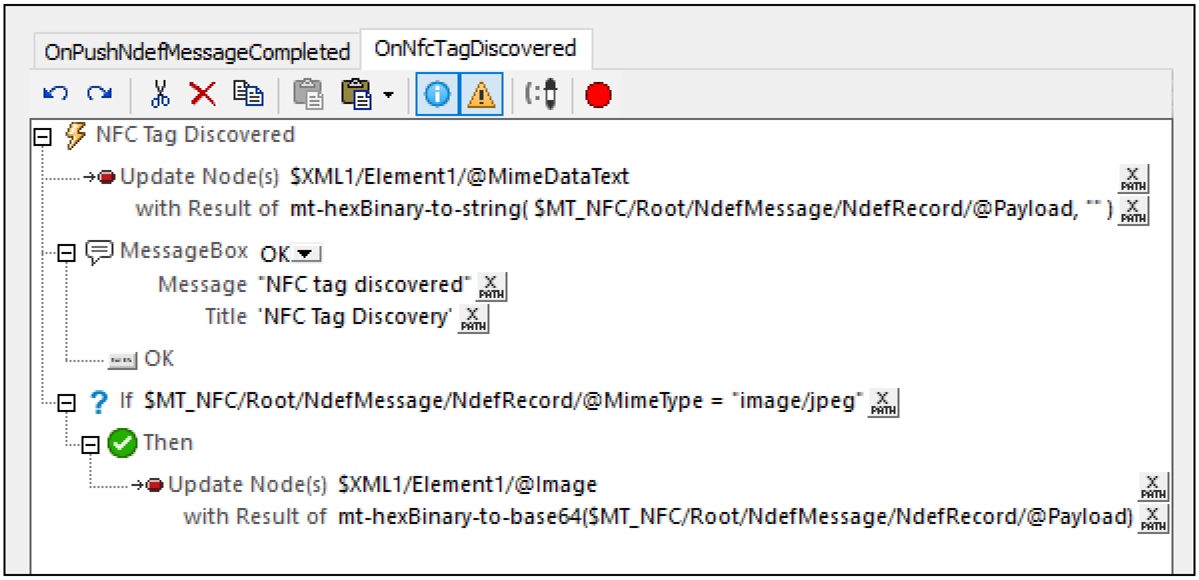
NFC Push Ndef Message Completed
MessageBox OK
Message "NFC message completed"
Title "NFC Push Action"
OK

See also Pushing Data to Other Devices[1120].

## OnNfcTagDiscovered

When an NFC tag is discovered[1119], the information in it is automatically read and stored in the `$MT_NFC` tree. This event enables you to specify any additional actions you might want to carry out. For example, as shown in the screenshot below, page-source nodes can be updated with data from the **`Payload`** attribute of the `$MT_NFC` tree's **`NdefRecord`** element.

Note that the **`Payload`** attribute will have its content in hexBinary format. If the payload is known to be carrying a text string, then the extension function[1262] `mt-hexBinary-to-string` can be used to obtain the text string before placing the string in a page-source node (*see the first Update Node*[886] *action in the screenshot below*). Similarly, if the payload is expected to carry an image, then the hexBinary content of the payload can be converted to a Base64 encoding of the image[1098] by using the `mt-hexBinary-to-base64` extension function.



NFC Tag Discovered
Update Node(s) $XML1/Element1/@MimeDataText
with Result of mt-hexBinary-to-string( $MT_NFC/Root/NdefMessage/NdefRecord/@Payload, "" )
MessageBox OK
Message "NFC tag discovered"
Title 'NFC Tag Discovery'
OK
If $MT_NFC/Root/NdefMessage/NdefRecord/@MimeType = "image/jpeg"
Then
Update Node(s) $XML1/Element1/@Image
with Result of mt-hexBinary-to-base64($MT_NFC/Root/NdefMessage/NdefRecord/@Payload)

See also Discovering and Reading NFC Tags[1119].

## 12.4.4 Overview of NFC Design Components

NFC functionality is implemented with the help of the following NFC-specific design components:

▼ NFC Start/Stop action

This action is used to start or stop the pushing and/or receiving of messages.

Pushing and/or receiving is started when the ***NFC Start*** 746 action is triggered. The sequence of steps that the action sets in motion is as follows:

1. NFC must be enabled on the device. If NFC is not enabled, then triggering the *Start* action will cause a prompt to appear asking the user to enable NFC.
2. After ensuring that NFC is enabled, the MobileTogether Client app will be registered for NFC.
3. Immediately thereafter, NFC tag discovery starts automatically and NFC messages in NFC tags will be received automatically. Pushing can be started on Windows devices via an NFC Push action 747; it is not automatically started.

The ***NFC Stop*** 746 action stops the pushing and receiving of all messages. To re-start the pushing and receiving of messages, trigger the *Start* action again.

See also Discovering and Reading NFC Tags 1119.

▼ NFC Push action

The NFC Push action 747 works on Windows devices and enables data to be transmitted from the Windows device running the solution to any other NFC-enabled device. The NFC Push action 747 defines the message or file to push. When the action is triggered, the specified message or file is transmitted via NFC.

**Note:** NFC data-pushing is supported on Windows devices, but not on Android or iOS devices.

For more information, see Pushing Data to Other Devices 1120 and NFC Push action 747.

▼ $MT_NFC page source tree

The `$MT_NFC` tree is automatically created as a page source in the design when an NFC Start 746 action is defined. The tree is automatically populated when an NFC tag is discovered. The data from the NFC tag is stored in the nodes of the `$MT_NFC` tree. For simulations, you can use an NFC sample file 1377 to see how data from its NFC tag is stored in the `$MT_NFC` tree. (See the section NFC Sample Files 1377 for more information about NFC simulations.)

▼ NFC-related events

Two NFC-related events provide crucial functionality:

- `OnPushNdefMessageCompleted` specifies what action/s to carry out when the transmission of NFC data (via NFC Push 747) has been completed.

- `OnNfcTagDiscovered` specifies what action/s to carry out when an NFC tag is discovered[1119]. For example, when this event is triggered, an Update Node[886] action can be used to update data-source trees with data from the discovered NFC tag.

## NFC-related XPath extension functions

The following NFC-related functions are available:

- `mt-nfc-started`: a Boolean test to check whether the solution has started NFC[746].
- `mt-hexBinary-to-string`: converts a hexBinary to a text string.
- `mt-hexBinary-to-base64`: converts a hexBinary to a Base64-encoded image.
- `mt-string-to-hexBinary`: converts a text string to a hexBinary string.
- `mt-base64-to-hexBinary`: converts a Base64-encoded image to a hexBinary string.

Since the payload of messages is transported in hexBinary format, the conversion functions enable data to be prepared for transport (that is, converted to hexBinary) and converted from hexBinary to human-usable formats (text and images). For more detailed descriptions of these functions, see MobileTogether Extension Functions[1262].

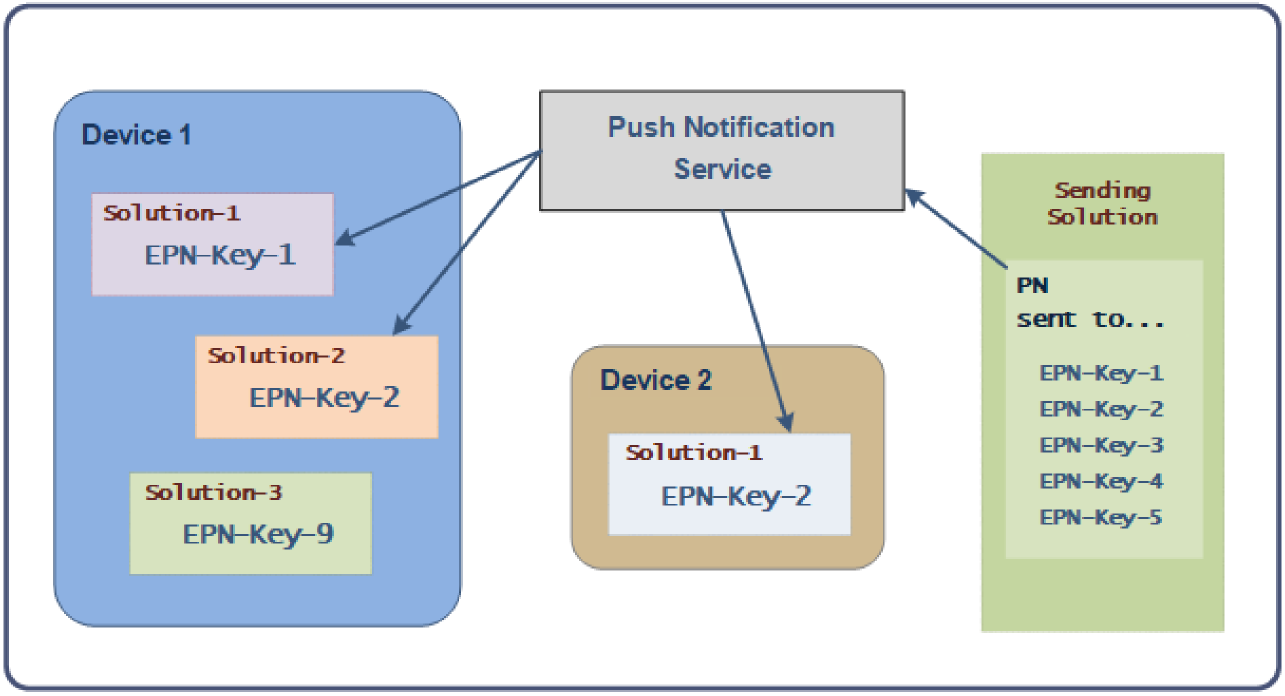
## NFC sample files for simulations

For simulations, you can create an NFC sample file[1377] and use this file to test whether data from NFC tags is being correctly imported into the `$MT_NFC` tree. See the section NFC Sample Files[1377] for more information about NFC simulations.

## 12.5 Push Notifications

A push notification (PN) is a text message that appears on a mobile device and that is related to an app that is installed on the device. When a PN is received on the device, the user does not need to be using the app or even using the device. All that is required is that the device be switched on. PNs thus provide a way for app publishers to communicate directly to users, without having to wait for the app to be started or to get lined up among incoming messages. PNs typically provide information, such as update news related to the app, but can also be used to drive actions, such as accepting invitations, linking to a website, or modifying a database.

In MobileTogether, a PN is sent from a MobileTogether solution on one device and is received by the same solution (or another solution) on other devices. The notification is thus "pushed out" from one solution to multiple devices. The screenshot below shows how a PN is sent from a sending solution to receiving solutions, with the recipients in this case being identified on the basis of an external PN key (other methods for identifying recipient devices are also available).



PNs can also be sent by MobileTogether AppStore Apps—[1471] which are MobileTogether solutions that have been compiled as apps that can be downloaded from app stores—to other MobileTogether AppStore Apps[1471].

### How MobileTogether push notifications work

MobileTogether PNs consist of a short message, a big message, and a payload consisting of data structured as key–value pairs. Typically, the short message is what is displayed on the device when the PN is received; on tapping the PN, the big message is displayed; the messages can have buttons, which allows the user to determine what action to take when the message is received. The payload of the PN is transferred to a data tree on the device and can be used by other actions; this enables new data—data that is related to the PN event—to be freely processed by the whole range of MobileTogether actions[667] and used with MobileTogether controls[403].

Broadly, the push notification mechanism works as follows:

- A Send Push Notification[753] action of the sending solution defines the PN's main parameters (message details and list of receivers). This part of the mechanism is broadly defined in The Sending Solution[1126] sub-section of this section.
- Receiving devices are identified on the basis of either their login credentials or their registration. A device is logged in to a particular MobileTogether Server with a specific **user name**. Additionally, a device can be registered (i) with **an external PN key**, and/or (ii) to receive specific **PN topics**. In the Send Push Notification[753] action, there is a setting that allows receiving devices to be defined on the basis of their user name, their external PN key, or the PN's topic. Because of this, a PN can be sent to a list of receivers that can be flexibly configured.
- When a PN is received on a device, its short/big messages can be displayed and its payload is automatically transferred to the `$MT_PUSHNOTIFICATION` page source of the receiving solution. The receiving solution defines what additional action/s to take when the PN is received and if the user taps one of the PN's buttons. The sub-section The Receiving Solution[1128] provides an overview of this part of the PN mechanism.

### In this section

While The Sending Solution[1126] and The Receiving Solution[1128] sub-sections together describe the main PN mechanism, the two sub-sections that follow them deal with additional features. In order to successfully implement PNs in AppStore Apps[1471] (apps that are compiled from MobileTogether solutions), some additional steps are required; these steps are described in the Push Notifications in AppStore Apps[1130] sub-section. Simulating Push Notifications[1133] explains how to simulate PNs when the sending and receiving solutions are not one and the same solution.

## 12.5.1 The Sending Solution

In the sending solution, you select an event that will trigger the sending of the push notification (PN), and then define the Send Push Notification[753] action for this event (*screenshot below*). The various parameters of the PN are defined in the Send Push Notification[753] action itself. The main parameters are briefly introduced below.

Additional to defining the parameters of the Send Push Notification[753] action, the only other preparations required in the sending solution relate to the data to be sent in the PN. If data for the payload and/or short and big messages is added dynamically from page sources of the sending solution, then these page sources must be created and the correct data made available for the PN's payload.

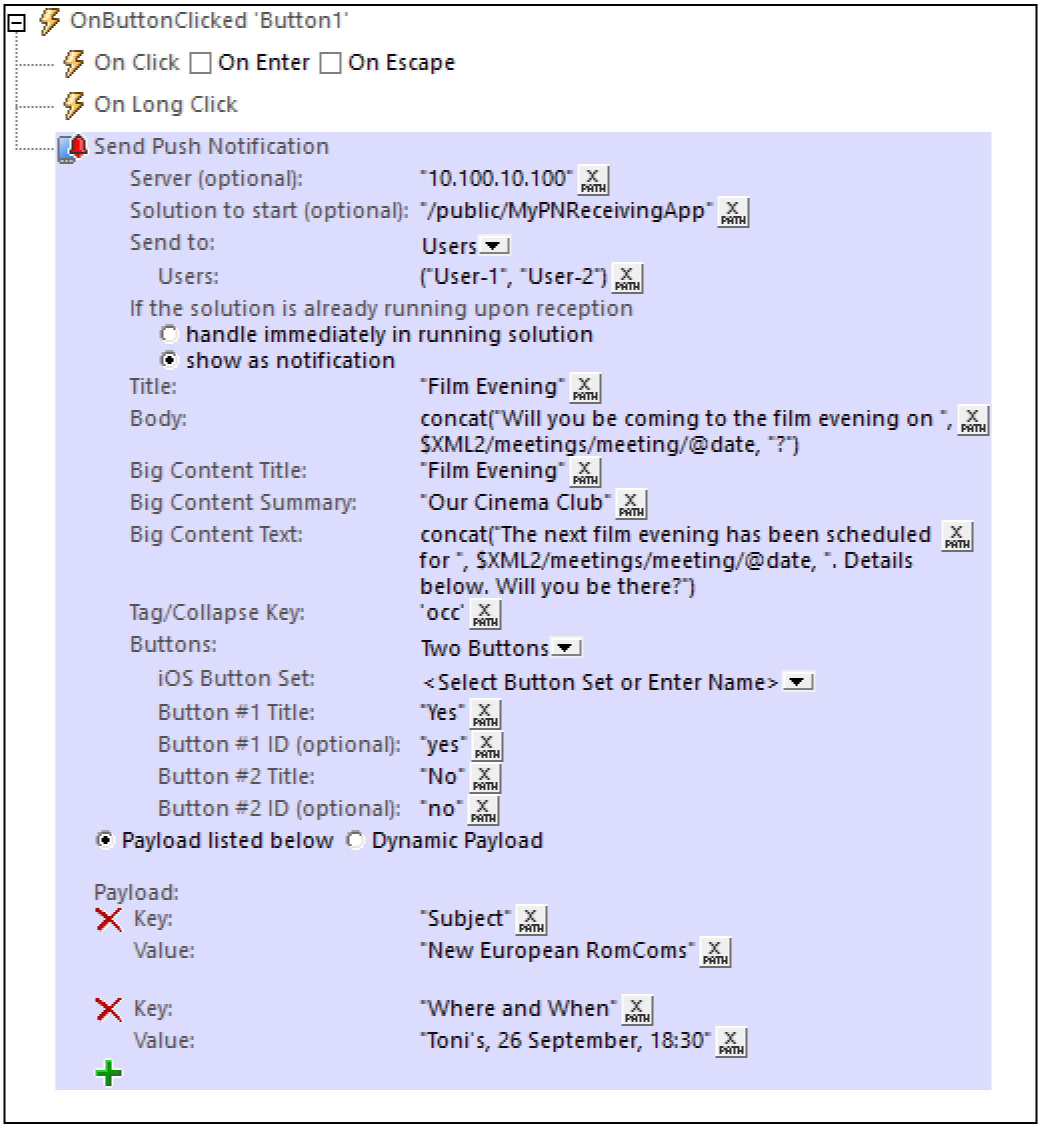
While the mechanisms listed below are part of the sending solution, note that the sending solution can be the same solution as the receiving solution. If this is the case, then the mechanisms of both the sending and receiving solutions are combined in one and the same solution.

### Preparing the sending solution: the Send Push Notification action

Most of the preparation in the sending solution concerns the definitions of the PN's parameters in the Send Push Notification[753] action. The main parameters are as follows:

- A short message, big message, and payload are defined. Defining these is straightforward.
- The recipient list is defined here, on the basis of user names (or user roles), external PN keys, and PN topics. A device is logged in to a particular MobileTogether Server with a specific user name. User

names and their roles refer to the users configured on MobileTogether Server. Additionally, a device can be registered (i) with an external PN key, and/or (ii) to receive specific PN topics. *(For more information about these registrations, see The Receiving Solution[1128].)* The recipient list can thus be flexibly configured using any one of these selection criteria.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Send Push Notification
Server (optional): "10.100.10.100"
Solution to start (optional): "/public/MyPNReceivingApp"
Send to: Users
Users: ("User-1", "User-2")
If the solution is already running upon reception
○ handle immediately in running solution
● show as notification
Title: "Film Evening"
Body: concat("Will you be coming to the film evening on ", $XML2/meetings/meeting/@date, "?")
Big Content Title: "Film Evening"
Big Content Summary: "Our Cinema Club"
Big Content Text: concat("The next film evening has been scheduled for ", $XML2/meetings/meeting/@date, ". Details below. Will you be there?")
Tag/Collapse Key: 'occ'
Buttons: Two Buttons
iOS Button Set: <Select Button Set or Enter Name>
Button #1 Title: "Yes"
Button #1 ID (optional): "yes"
Button #2 Title: "No"
Button #2 ID (optional): "no"
● Payload listed below ○ Dynamic Payload

Payload:
Key: "Subject"
Value: "New European RomComs"

Key: "Where and When"
Value: "Toni's, 26 September, 18:30"

- A PN can contain PN buttons. You can define the text and ID of up to three PN buttons. When a user taps a button, the button's ID is passed to a page source node of the solution. This provides a way to determine how the user wishes to react to the PN, and consequently enables corresponding actions to be defined (in the tab of the **`OnPushNotificationReceived`** event). Note that, PN buttons that are displayed on iOS devices are enabled for AppStore Apps[1471], not for standard MobileTogether solutions.
- If no button is defined and the user taps the PN, then no button ID is passed to the page source and the actions defined in the tab of the **`OnPushNotificationReceived`** event are executed.

### The sending solution as an AppStore App

If the sending solution is being created as an AppStore App[1471], please see Push Notifications in AppStore Apps[1130] for additional steps to take

## 12.5.2 The Receiving Solution

This section lists the different parts of the Push Notification (PN) mechanism that are present in the receiving solution. While the mechanisms listed below are part of the receiving solution, note that the receiving solution can be the same solution as the sending solution. If this is the case, then the mechanisms of both the receiving and sending solutions are combined in one and the same solution.

**Note:** The Big Content of standard MobileTogether solutions is displayed only on Android and Windows devices. If you want Big Content to be displayed on iOS devices, then compile the receiving solution as an AppStore App[1130].

### OnPushNotificationReceived event

- *At design time*, the **`OnPushNotificationReceived`** event is accessed via the receiving solution's project properties[296]. In the event's tab, you define the actions to carry out when the PN is received. When an action is added to this event at design time, the `$MT_PUSHNOTIFICATION` page source is automatically created.
- *At run time*, the **`OnPushNotificationReceived`** event is triggered when the user taps the PN or a button in the PN. When the event is triggered, the following happens: (i) the receiving solution is started if it is not already running; (ii) the PN's payload is automatically transferred to the `$MT_PUSHNOTIFICATION` page source; if a PN button was pressed, then, additionally, the button's ID (which is a string) is passed to the page source; (iii) the event's actions will be executed; note that, by using the If-Then[894] or If-Then-Else[894] action, actions can be made dependent on what PN button was pressed (*see next section below*).

### $MT_PUSHNOTIFICATION page source

The `$MT_PUSHNOTIFICATION` page source has the following fixed structure:

```
$MT_PUSHNOTIFICATION
Root
|   @button
|
|-- Entry
|      @key
|      @value
```

At run time:

- If a PN button was tapped, then the button's ID is passed to the `$MT_PUSHNOTIFICATION/Root/@button` node. The value of the **`@button`** attribute can be used for conditional processing by using the If-Then[894] or If-Then-Else[894] action. For example, if the **`@button`** node contains an *Accept* button's ID, then an acceptance SMS can be automatically sent or a database suitably modified; alternative actions can be defined for other button IDs.

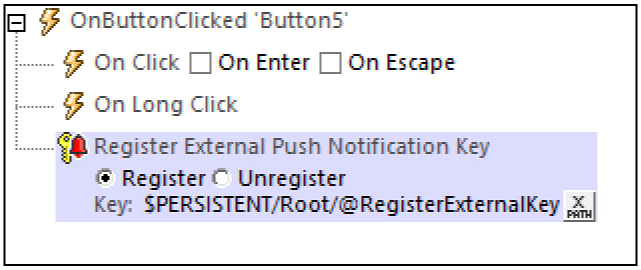
- The number of **`Entry`** elements is determined at run time and will be equal to the number of key–value pairs contained in the PN's payload. The data of each key–value pair will be passed to a corresponding **`Entry`** element. Data in the `$MT_PUSHNOTIFICATION` page source can be processed in any way that you want, including to simply display the data in the design.

## External PN keys

A PN-receiving mobile device can be registered by using *one external PN key per solution*. This key is a text string that is generated by the Register Ext PN-Key 757 action (*see screenshot below*). If a PN is sent to a given external PN key, then all devices that are registered with that key will receive the PN.

The fact that one PN key can be registered per solution has two implications:

- The device can be registered with different external PN keys, but each PN key is tied to a specific receiving solution. If two solutions on the device use the same PN key, then a push notification that targets this key will be delivered to both solutions on that device.
- Since the same key can be generated by the same solution on other mobile devices, the external PN Key serves to identify a particular set of mobile devices. If a push notification is sent to a particular PN key, then devices that were registered with this PN key will receive the push notification.

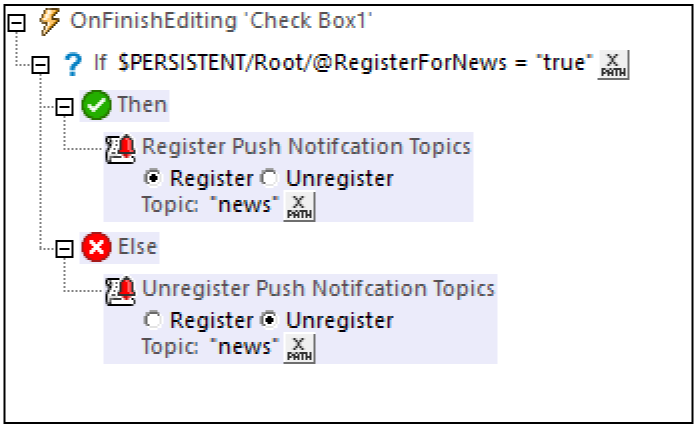


OnButtonClicked 'Button5'
On Click □ On Enter □ On Escape
On Long Click
Register External Push Notification Key
● Register ○ Unregister
Key: $PERSISTENT/Root/@RegisterExternalKey

**Note:** External PN keys can also be used in AppStore Apps 1471..

## PN topics

A PN-receiving mobile device can be registered to receive PNs about one or more specific topics. A device is registered for one or more topics via the Register PN-Topics 759 action (*see screenshot below*). At run time, if a PN is sent to a given PN topic, then all devices that are registered for that topic will receive the PN. If a PN is sent to multiple PN topics, then devices that are registered for any of the target topics will receive the PN.



```
⊟ OnFinishEditing 'Check Box1'
  ⊟ ? If $PERSISTENT/Root/@RegisterForNews = "true"
    ⊟ Then
        Register Push Notifcation Topics
          ● Register ○ Unregister
          Topic: "news"
    ⊟ Else
        Unregister Push Notifcation Topics
          ○ Register ● Unregister
          Topic: "news"
```

In theory, any solution on a particular device can be used to register that device for a given topic. In practice, it is best that the registration for a topic be done from the solution that will be receiving the PN.

### iOS PN button set definitons

When a PN containing a PN button arrives on a device and the button is tapped, then the receiving solution is started and the PN button's ID is passed to the `$MT_PUSHNOTIFICATION/Root/@button` node of the solution's `$MT_PUSHNOTIFICATION` page source. This is all that the PN button does. In effect, it provides a way to determine how the user wishes to react to the PN.

While PN buttons for non-iOS devices are defined in the Send Push Notification 753 action of the sending solution, iOS PN button sets are defined in the receiving solution via the **Project | iOS Push Notification Button Sets** 1608 command.

### The receiving solution as an AppStore App

If the receiving solution is being created as an AppStore App 1471, please see Push Notifications in AppStore Apps 1130 for additional steps to take.

## 12.5.3 Push Notifications in AppStore Apps

AppStore Apps 1471 are MobileTogether apps that are compiled into different OS-based apps. These apps can be posted to the respective app stores for end users to download. (See the section AppStore Apps 1471 for details.) In order to compile the program code for the different operating systems, you must first generate the program code for the respective operating systems. This is done with the help of the Generate Program Code Wizard 1472. If your MobileTogether app of the **receiving solution** is set to receive push notifications (PNs), then you must fill in the relevant settings in the Wizard's Screen 6. (Note that this screen appears only if the **`OnPushNotificationReceived`** 296 event of the solution has an action defined for it.)

Using PNs in AppStore Apps 1471 entails some additional steps, which are explained below. You will need to carry out these steps before generating the program code; that's because the registration information you will obtain after carrying out these steps is required in order to complete the settings in Screen 6 of the Generate Program Code Wizard 1472.

**Note:** The steps to prepare an app to receive push notifications apply only to the **apps of receiving solutions**. If the apps for the sending and receiving solutions are different apps, then the steps outlined in this section apply only to the app of the receiving solution; they do not apply to the app of the sending solution.

## Android and iOS

In order for your Android and iOS apps to receive PNs, you must have a Firebase account and have created a Firebase Server key.

Note down the Firebase Server key for use in Screen 6 of the Generate Program Code Wizard 1472.

## Android

Create your Android app in the Firebase Console. Go to the app's page and download its `google-services.json` file.

Note the location of the downloaded `google-services.json` file. Do not change the name of the file. You will need to point to this file in Screen 6 of the Generate Program Code Wizard 1472.

## iOS

You will need to: (i) generate an *Apple Push Notification service (APNs) key*, and (ii) do some configuration in Firebase, and generate and download a `GoogleService-Info.plist` file. To do these, carry out the steps listed below.

Generate an APNs key *(needed only one per Apple developer account; can be used across various projects)*:

1. Create an APNs key at **`https://developer.apple.com`**. Do this as follows: Under *Account > Certificates, ID & Profiles > Keys*, click the **Add** button (`+`), and create the key.
2. Confirm and download the APNs key file, which has a `.p8` extension and is generated once only. Store the key file at a secure location. Also store the 10-digit authentication key ID. The key file and the authentication ID are needed only once, to set up for your Apple Developer Account.

Configure Firebase, and generate and download a `GoogleService-Info.plist` file:

1. Create your iOS app in your Firebase Console.
2. Upload the `.p8` file that you generated earlier.
3. Specify app id prefix—not bundle ID prefix—as well as the 10-digit authentication key id (obtained during APNs key generation).
4. Download the `GoogleService-Info.plist` file.

Note the location of the downloaded file. Do not change the name of the file. You will need to point to this file in Screen 6 of the Generate Program Code Wizard 1472.

*Troubleshooting*

**Q:** *What might be the problem if push notifications are not received on the phone?*
**A:** *One or more of the following issues might be responsible:*

- After the AppStore App has been installed, it must be started at least once. When asked if receiving PNs should be allowed, click **Yes**. After the app starts and contacts the server, the client's PN address is sent to the server.

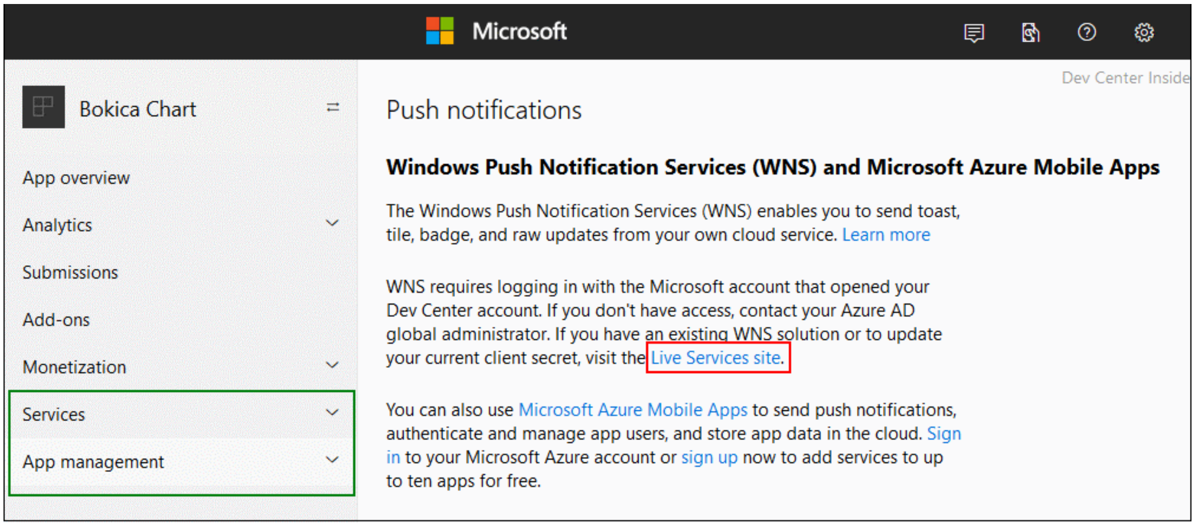
- If you can receive broadcast messages for a specific topic but not for a compiled app, then check that the solution name to which the PN is sent is correct.
- Check that the **`p8`** file and the two keys (APNs and authentication key) have been uploaded.
- Check that the Bundle ID in the Firebase settings and in the generated AppStore App are identical.
- Check that PNs for your app ID in Apple's Developer Console have been enabled.
- If none of the above apply, go to **`xcode/your project/capabilities`**. Switch off the PN switch, and then switch it on again.

**Q:** *"Show as notification" does not work on iOS 9.*
**A:** This is not an error. Support is provided for iOS 10 or later.

## Windows

When generating program code for a Windows app of a PN-receiving solution, you will need, for Screen 6 of the Generate Program Code Wizard[1472], two pieces of information: (i) the Package SID, and (ii) the Application Secret. The procedure to obtain these is as follows:

1. Login at https://developer.microsoft.com/en-us/dashboard with your user name and password.
2. If the app does not yet exist in the store, you can proceed with the steps that follow after reserving a name for the app.
3. Go to your app's page, and there select **`Services > Push notifications > WNS/MPNS`**; in some cases, the **`WNS/MPNS`** item is listed under **`App management`** (*see screenshot below*).



Push notifications

**Windows Push Notification Services (WNS) and Microsoft Azure Mobile Apps**

The Windows Push Notification Services (WNS) enables you to send toast, tile, badge, and raw updates from your own cloud service. Learn more

WNS requires logging in with the Microsoft account that opened your Dev Center account. If you don't have access, contact your Azure AD global administrator. If you have an existing WNS solution or to update your current client secret, visit the Live Services site.

You can also use Microsoft Azure Mobile Apps to send push notifications, authenticate and manage app users, and store app data in the cloud. Sign in to your Microsoft Azure account or sign up now to add services to up to ten apps for free.

4. In the page that appears, click *Live Services site* (*boxed in red in the screenshot above*).
5. The Registration page of the app will open. Note down the following details from this page; you will need them to generate the program code with the Generate Program Code Wizard[1472]: (i) Package SID, (ii) Application Secrets. Click **Save** to finish the procedure.
6. You can now start the Generate Program Code Wizard[1472] to create the program code.

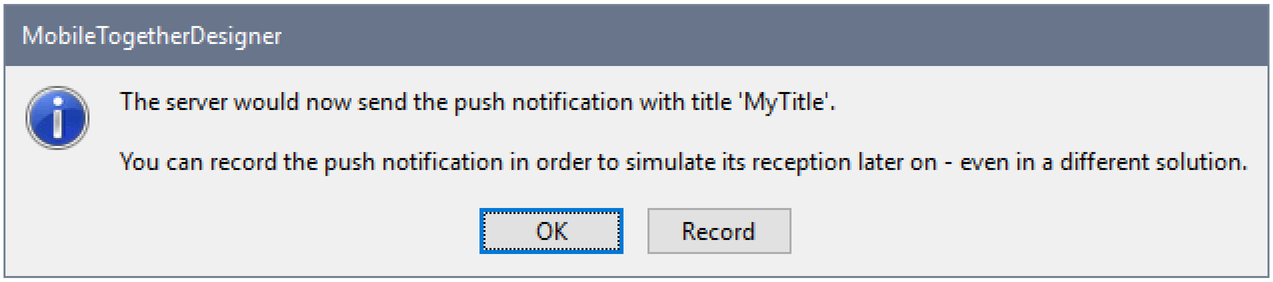
## 12.5.4 Simulating Push Notifications

A push notification (PN) contains data related to: (i) the PN's short message, (ii) the PN's big message, and (iii) the PN's payload. If the sending and receiving solutions are one and the same solution, then, in a simulation, the data transfer between sending and receiving parts is carried out within that one solution's simulation; the simulation in this case is straightforward.

However, if the sending solution and receiving solution are different, then the simulation mechanism is as follows: PN data sent during a simulation of the sending solution is recorded in a MT PN Simulation file (which has a `.mtpnsim` extension). When the receiving solution is simulated, you can load that `.mtpnsim` file. The simulator will now display all the sets of PN data in the `.mtpnsim` file, and you can select the PN that you want to simulate.

### Recording PN Simulation Data

If, while simulating a sending solution, you trigger an event that sends a PN, then the dialog shown below appears.



The server would now send the push notification with title 'MyTitle'.

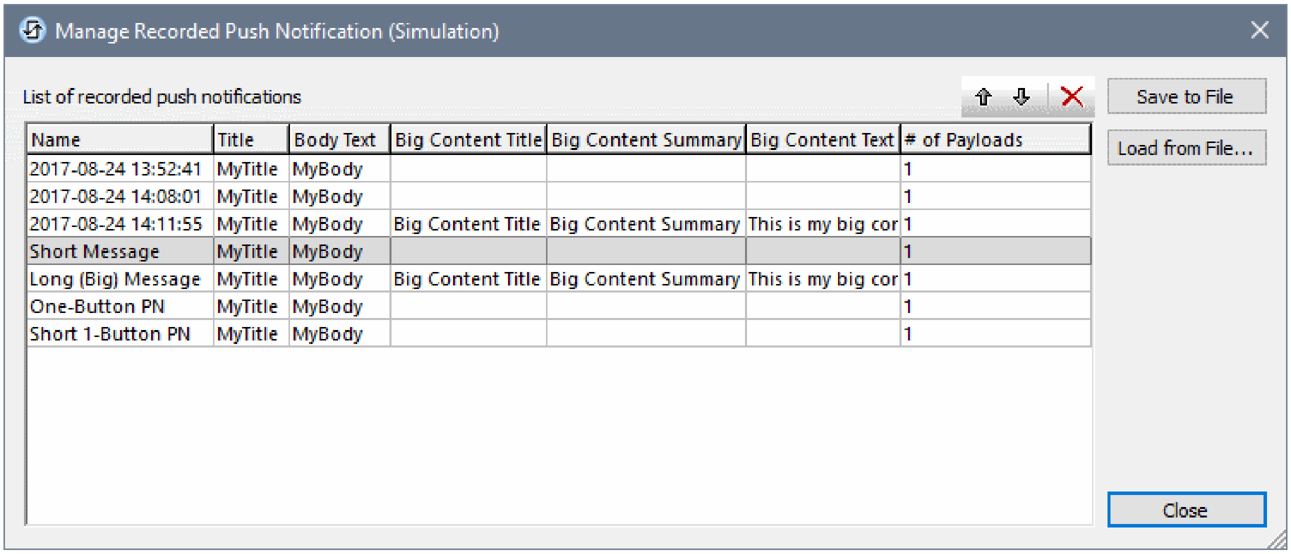
You can record the push notification in order to simulate its reception later on - even in a different solution.

Click **Record** to record the PN data to memory. You can record multiple PNs to memory in this way. On closing the simulation, the recorded PNs are in memory—but not yet saved to file.

When you save or close the solution file, you will be informed that there is unsaved recorded PN data in memory, and you will be asked whether you wish to save this data to file. If you select **Yes**, then the different sets of recorded PN data in memory will be saved to a MT PN Simulation file in the same folder as the solution. This file will be named with the following pattern: `YourSolutionName.mtpnsim`. Any additional PNs sent during the current or subsequent simulations of that solution will be saved to the same file. Each set of PN data in the file is identified by a name, which is that PN's recording date and time.

### Loading recorded PN Simulation Data into the simulator

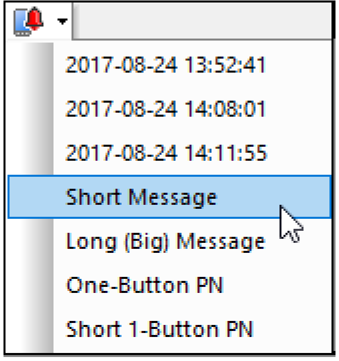
To load recorded PN simulation data from an MT PN Simulation file (`.mtpnsim` file), start a simulation of the receiving solution (an MTD file) and then click the **Push Notifications** icon 🔔▾ in the simulator's toolbar. This causes the Manage Recorded Push Notification (Simulation) dialog (*see screenshot below*) to appear. Click **Load from File**, then browse for the `.mtpnsim` file you want to load, and click **Open**. The recorded PN data in this file will be loaded into the dialog (*see screenshot*) and into memory. It will not automatically be saved to the MTD file. If you reload the MTD file without saving, then the recorded PN data will have to be reloaded from the `.mtpnsim` file. Click **Save to File** to save the recorded PN data to the MTD file; this will prevent you having to reload the `.mtpnsim` file. If you load PN data from another `.mtpnsim` file, then the new data will overwrite the PN data in memory. If you now wish to overwrite any recorded PN data in the MTD file, click **Save to File** in this dialog.


Manage Recorded Push Notification (Simulation)
List of recorded push notifications
Save to File
Load from File...
Close

| Name | Title | Body Text | Big Content Title | Big Content Summary | Big Content Text | # of Payloads |
|---|---|---|---|---|---|---|
| 2017-08-24 13:52:41 | MyTitle | MyBody | | | | 1 |
| 2017-08-24 14:08:01 | MyTitle | MyBody | | | | 1 |
| 2017-08-24 14:11:55 | MyTitle | MyBody | Big Content Title | Big Content Summary | This is my big cor | 1 |
| Short Message | MyTitle | MyBody | | | | 1 |
| Long (Big) Message | MyTitle | MyBody | Big Content Title | Big Content Summary | This is my big cor | 1 |
| One-Button PN | MyTitle | MyBody | | | | 1 |
| Short 1-Button PN | MyTitle | MyBody | | | | 1 |

In the Manage Recorded Push Notification (Simulation) dialog (*screenshot above*), each recorded PN is shown on a separate line, and is shown with its name, short message data, big message data, and payload information. You can change the order of the PNs by selecting one or more of them and clicking the **Move Up** and **Move Down** toolbar icons (located top right). You can delete a PN by selecting it and clicking the **Delete** toolbar icon. You can also edit the names of PNs so that you can identify them more easily; to do this, double-click the name and edit. Changes made in the dialog are made in memory. To save the changes to the MTD file, click **Save to File**.

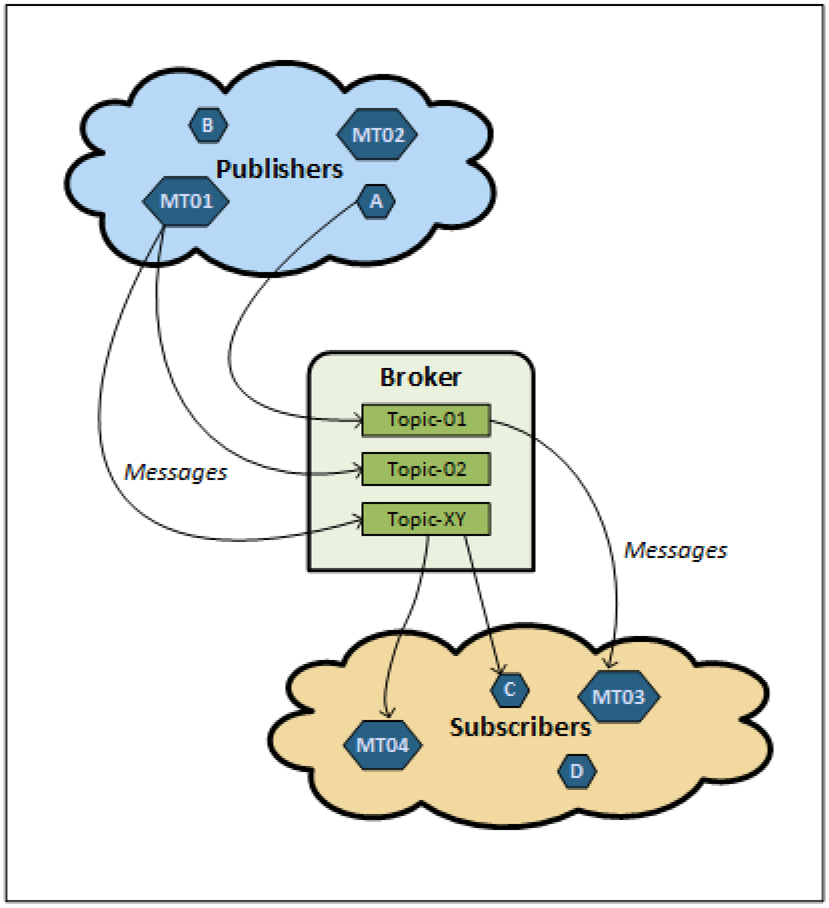
To select which PN is used in the simulation, click the dropdown arrow of the **Push Notifications** icon in the simulator's toolbar. This displays a list of all the PNs that are currently in memory (*see screenshot below*). The order in which PNs are displayed is the same as their current order in the Manage Recorded Push Notification (Simulation) dialog (*compare screenshot below with that above*).



After you select a PN from this list, the solution will simulate that it has received the selected PN. To simulate the receipt of another PN, select a new PN from the dropdown list.

# 12.6 MQTT

MQ Telemetry Transport (MQTT) is a lightweight messaging protocol that uses a publish-and-subscribe model that is intermediated by a broker. A publisher publishes messages under a given topic name to a broker. A subscriber to that topic at the broker will receive messages published under that topic name. For example, a publisher may be a temperature sensor in an agricultural field that sends a temperature reading (publishes the reading) to Broker Z at a particular time under a topic name of, say, *MaizyWheatField-123A*. An irrigation system computer that is subscribed to the topic *MaizyWheatField-123A* at Broker Z will receive the published temperature readings and can be programmed to switch on/off the irrigation system according to the latest published temperature.



MobileTogether supports MQTT by enabling MT solutions to join an MQTT network as a publisher, or as a subscriber, or as both. For example, in the diagram above, the MT solution named **MT01** publishes messages to two topics (*Topic-02* and *Topic-XY*), the MT solution named **MT03** subscribes to *Topic01*, which is published by Publisher A, and the MT solution named **MT04** subscribes to *Topic-XY*. Note that an MT solution can be both a publisher and a subscriber at the same time.

**Note:** An MT solution that is a subscriber can subscribe to topics published by both MobileTogether publishers as well as non-MobileTogether publishers.

In the topics of this section, we describe the following:

- How to set up a solution to publish messages, subscribe and unsubscribe to topics, and disconnect from a broker 1136
- How to specify the solution actions to perform when the solution receives an MQTT message 1138
- How to set up an MQTT service 1139
- How to run a simulation 1140 to test the action tree that is executed when an MQTT message is received

### MQTT specification and links

The documentation in this section explains how to set up your MobileTogether solution to publish, and subscribe to, MQTT messages and how to define actions to perform when an MQTT message is received. For more information about MQTT, see the following resources:

- the MQTT website
- the specification of MQTT Version 3.11, which is the version supported in MobileTogether

## 12.6.1 Publish, Subscribe, Disconnect Broker

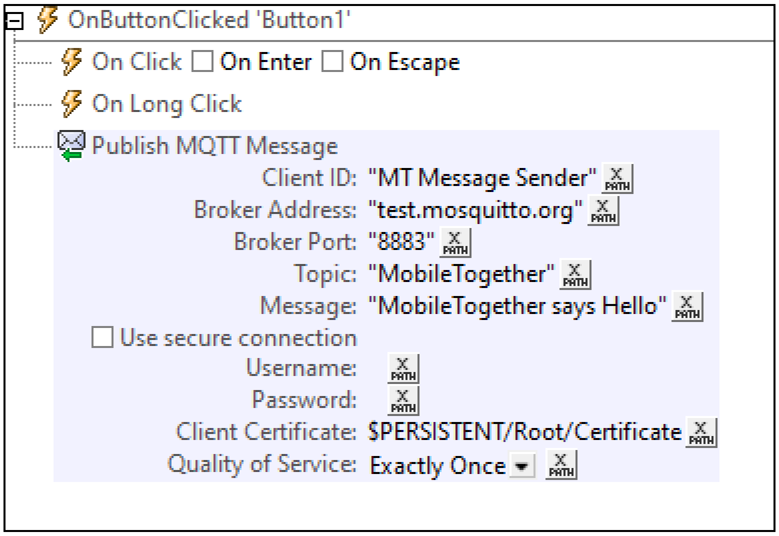
A MobileTogether solution can participate in an MQTT network as a publisher, a subscriber, or both. This topic describes the mechanisms that set up a solution as a publisher and as a subscriber, and it describes how to disconnect a solution from a broker.

### Set up as a publisher

In order to publish an MQTT message to a broker, the solution will need the following information:

- Information to connect to the broker
- The topic under which a message will be published
- The text of the message

You can specify this information in the Publish MQTT Message 763 action (*screenshot below*), which is described in the Publish MQTT Message 763 topic. Information for connecting to the broker is given in the *Broker Address* and *Broker Port* settings. The *Topic* and *Message* settings contain the other two important pieces of data needed to publish messages to the broker. If the connection is to be a secure connection, you will need to provide the additional information required to make the secure connection.


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Publish MQTT Message
Client ID: "MT Message Sender"
Broker Address: "test.mosquitto.org"
Broker Port: "8883"
Topic: "MobileTogether"
Message: "MobileTogether says Hello"
□ Use secure connection
Username:
Password:
Client Certificate: $PERSISTENT/Root/Certificate
Quality of Service: Exactly Once

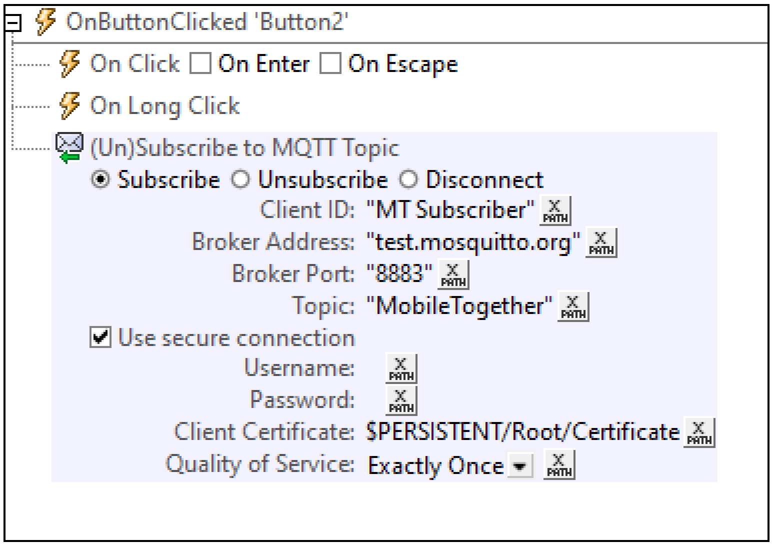
At run time, when the Publish MQTT Message 763 action is triggered (for example, on clicking a button in the solution), the solution will automatically attempt to connect to the broker. On connecting successfully, it will publish the message to the broker.

## Set up as a subscriber

In order to subscribe to an MQTT topic at an MQTT broker, the solution will need the following information:

- Information to connect to the broker
- The topic to which you want to subscribe
- The quality of service that you want for this subscription

You can specify this information in the Subscribe to MQTT 764 action (*screenshot below*), which is described in the (Un)Subscribe to MQTT 764 topic. Information for connecting to the broker must be given in the *Broker Address* and *Broker Port* settings. The topic you want to subscribe to is given in the *Topic* setting. You can specify the number of times individual messages should be delivered in the *Quality of Service* setting. If the connection is to be a secure connection, you will need to provide the additional information required to make the secure connection.



OnButtonClicked 'Button2'

On Click □ On Enter □ On Escape

On Long Click

(Un)Subscribe to MQTT Topic

● Subscribe ○ Unsubscribe ○ Disconnect

Client ID: "MT Subscriber"

Broker Address: "test.mosquitto.org"

Broker Port: "8883"

Topic: "MobileTogether"

☑ Use secure connection

Username:

Password:

Client Certificate: $PERSISTENT/Root/Certificate

Quality of Service: Exactly Once

At run time, when this action is triggered (for example, when the page is loaded in the solution), the solution will automatically attempt to connect to the broker. On successfully connecting, it will receive new messages that were published under the subscribed topic. New messages will be received by the solution and placed in the solution's **$MT_MQTT** 1138 page source. If more than one message is available, then the messages are queued for placement in the **$MT_MQTT** 1138 page source. After the action handling for a received message 1138 has been completed, the next message in the queue will be placed in the **$MT_MQTT** 1138 page source and its action handling will be started.

**Note:** After a Subscribe action establishes a connection to a broker, the connection will be kept alive till it is explicitly disconnected (*see next section below: Disconnect from broker*). You should this in mind if your solution subscribes to a large number of brokers and topics.

**Note:** You can also unsubscribe from a topic subscription via the Unsubscribe to MQTT 764 action. After the Unsubscribe action is executed, the solution will no longer receive messages about the topic from which it was unsubscribed.

### Disconnect from broker

When the actions to publish or subscribe are started (*see the description of the two actions above*), the solution will automatically connect to the broker using the connection information you have given. The connection to the broker will be kept alive in both cases till explicitly disconnected via the Disconnect from Broker 764 action. See the (Un)Subscribe to MQTT 764 topic for details of this action.

## 12.6.2 Actions on Message Received

After a solution has been subscribed to a topic at a broker, it will receive messages published by the broker under that topic. The message data (of the last message received) will be placed in the `$MT_MQTT` page source and can be accessed from there. The actions to be performed when a message is received is set on events at one of two levels: (i) an event at the page level 399 for each page; (ii) an event at the solution (project) level 296 (accessed via the Project Properties 296 section of the Styles & Properties pane 274). If no event handling is defined on the page-level event of the active page, then the actions of the project-level event will be executed.

In this topic, we describe the following:

- The `$MT_MQTT` page source
- The events, one each at page and solution level, that are triggered when a message is received.

## $MT_MQTT page source

The `$MT_MQTT` page source has the following fixed structure:

```
$MT_MQTT
Root
|-- Message
|       @topic
|       @content
```

At run time, if a solution has been subscribed to a topic at a broker, then that solution will receive messages published by the broker under that topic. The number of message repeats received by the solution is determined by the the *Quality of Service*[1136] level specified in the subscription.

The latest message that is received will be placed in the `$MT_MQTT` page source, with the body of the message going into the `$MT_MQTT/Root/Message/@content` node and the topic name going into the `$MT_MQTT/Root/Message/@topic` node. The message data in the page source will now be accessible to the solution and can be used for data processing.

**Note:** The content of every received message is automatically converted by MobileTogether to a hexBinary string and stored in this format in the `$MT_MQTT/Root/Message/@content` node. In order to convert the hexBinary string to a text string, use the `mt-hexBinary-to-string`[1262]. MobileTogether provides two functions to convert between hexBinary strings and text strings: `mt-hexBinary-to-string`[1262] and `mt-string-to-hexBinary`[1262].

## Events triggered when a message is received

When a solution receives a message, the `OnMQTTReceive`[399] event is triggered. You can specify what actions are to be performed when this event is triggered.

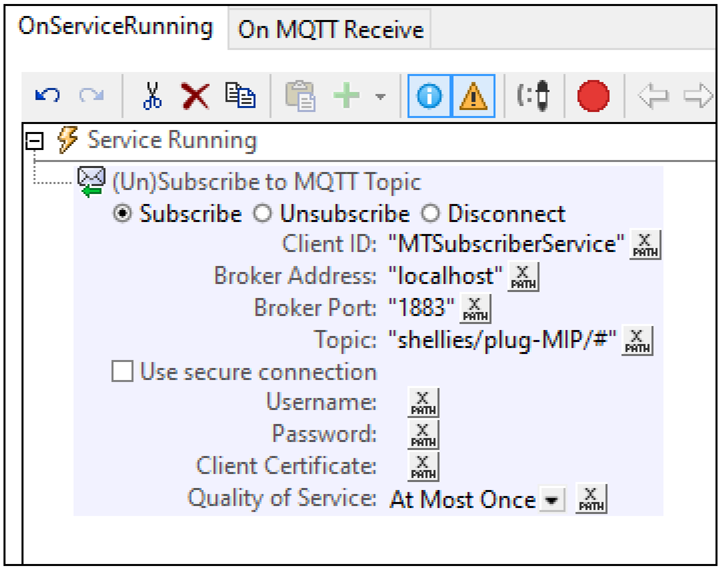
- If `OnMQTTReceive`[399] event actions have been defined at the page level[399], then these actions are executed when this page is active and receives a message.
- If no `OnMQTTReceive`[399] event action has been defined at the page level, then any `OnMQTTReceive`[399] event actions defined at the project level[296] will be executed.

This mechanism enables you to set actions at the project level, which can be triggered for any active page that does not have its own page-level `OnMQTTReceive`[399] event actions.

# 12.6.3 MQTT Service

You can run MQTT actions[761] as a service[1543] so that the actions run in the background on MobileTogether Server (Advanced Edition only). For example, you could set up a service to subscribe to an MQTT topic (*see screenshot below*) when some condition on the server is met, such as a time trigger. When a message is received on MobileTogether Server, the service is loaded and the action tree for received MQTT messages is executed.

The screenshot below shows the sequence of actions that will be performed when the service is started. These actions are defined in the `OnServiceRunning` tab. For MQTT services, you can use the additional `OnMQTTReceive` tab to set the actions to carry out when a subscription message is received.

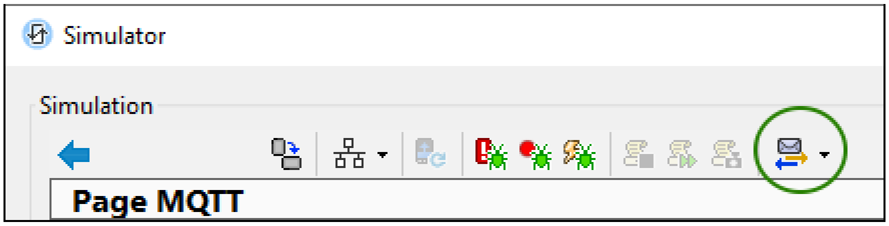


Service Running
(Un)Subscribe to MQTT Topic
● Subscribe ○ Unsubscribe ○ Disconnect
Client ID: "MTSubscriberService"
Broker Address: "localhost"
Broker Port: "1883"
Topic: "shellies/plug-MIP/#"
□ Use secure connection
Username:
Password:
Client Certificate:
Quality of Service: At Most Once

For more information about services and their triggers, see the section Server Services[1543].

## 12.6.4 Simulation

You can simulate what happens when an MQTT message is received. Press **F5** to start the simulator (*screenshot below*).



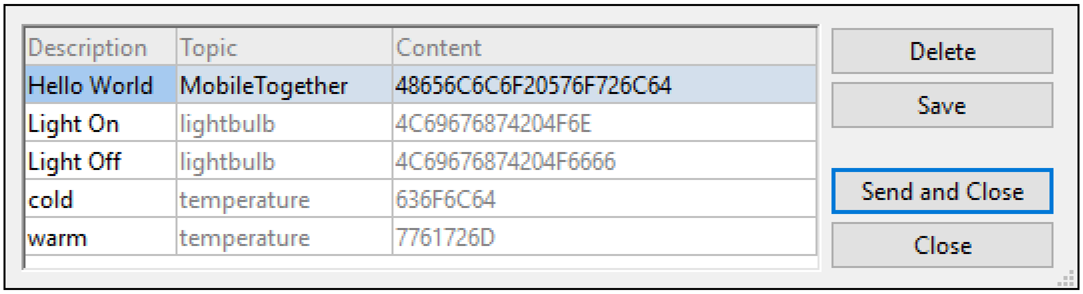
To simulate messages received, we use an **XML sample file** that is located by default in your MobileTogether Designer application folder. See the Options dialog (Simulation 2 tab)[1669] for the file path to this file. This file contains XML data that simulates messages received. So you can use it to test the action tree that will be executed when a message is received.

**Note:** If the XML sample file is in a write-protected folder (which is often the case with Windows application folders), you might need to open MobileTogether Designer with administrator privileges. This would enable you to save changes to the sample data (about which see below). To start MobileTogether Designer as an administrator, right-click its shortcut in Windows and select the command to start it as an administrator.

Start the MobileTogether Designer simulator by pressing **F5**. In the simulator[1355], use the *Simulate Reception of Message* toolbar button (*circled green in the screenshot above*) to open the Messages dialog.

In the dialog that appears, you can do the following:

- In the Messages dialog (*screenshot below*), the data displayed in the dialog's table is the data in the XML sample file. Each row represents one received message. The *Content* column contains the text of the message.

  | Description | Topic | Content |
  |---|---|---|
  | Hello World | MobileTogether | 48656C6C6F20576F726C64 |
  | Light On | lightbulb | 4C69676874204F6E |
  | Light Off | lightbulb | 4C69676874204F6666 |
  | cold | temperature | 636F6C64 |
  | warm | temperature | 7761726D |

  To simulate the reception of a message in the solution, select a message row and click **Send and Close**. The action tree of the solution will be executed with the data of the selected message and you can check whether actions were executed as you wanted. In the Messages dialog (*screenshot above*), you can edit the descriptions of messages and also delete a selected message. If you click **Save** after making changes, then the changes will be saved back to the XML sample file.

- The messages from the sample XML file are also available in the dropdown list of the *Simulate Reception of Message* toolbar button. Select a message in the dropdown list to simulate its receipt. Additionally, you can record a new message by first toggling on *Record a message* in the dropdown list of commands and then simulating the sending and receipt of a message. Whatever messages are sent while the recording is toggled on will be displayed in the Messages table (*screenshot above*). You can modify the descriptions as required and save the messages back to the XML sample file.

# 12.7 Broadcasts

MobileTogether's Broadcasts feature enables one or more solutions to broadcast messages to a topic. These messages are delivered via MobileTogether Server to all connected MobileTogether solutions that have been subscribed to this topic. Note that only MobileTogether solutions can participate in broadcasts. This is different than MQTT[1135], where messages are published via an MQTT broker and any device—not only MobileTogether solutions—can participate.

## Publish, subscribe

Publishing and subscribing is straightforward and each is achieved by triggering its respective action:

- To publish a broadcast message, the Publish Broadcast Message[768] action is defined. It has two settings: (i) the topic under which messages are broadcast; (ii) the text of the message to broadcast. When the action is triggered, the message is broadcast and will be delivered to all connected subscriber solutions.
- To receive a broadcast message, a solution must subscribe to the relevant topic—defined in the Subscribe Broadcast Topic[768] action. After the *Subscribe* action is triggered, the solution will start receiving messages published to the subscribed topic.
- The Unsubscribe Broadcast Topic[768] action enables you to stop a solution receiving messages of the specified topic. Note that the receipt of messages will also stop if a solution is no longer connected to MobileTogether Server. In case a solution stops receiving the messages of a given topic—either because of an *Unsubscribe* action or a lost connection—then a *Subscribe* action to that topic must be triggered for the solution to again receive messages from that topic.

**Note:** Broadcast messages must be strings.

## Actions on message received

When a broadcast message is received in a solution, (i) the message is stored in the solution's `$MT_Broadcast` dynamic variable[1304] and (ii) the action tree of `OnBroadcastReceive` is executed.

A **`$MT_Broadcast`** variable is declared for each received broadcast message. It contains the text of the received message and can be used in the action tree of that received message. The message held in **`$MT_Broadcast`** cannot be accessed after the `OnBroadcastReceive` action tree has finished executing. So if you want to use the received message at a later time, you should add an action to the `OnBroadcastReceive` action tree that stores the received message in a page source node. After the action tree has been executed, the **`$MT_Broadcast`** variable will no longer exist. A new **`$MT_Broadcast`** variable will be declared when the next broadcast message is received.

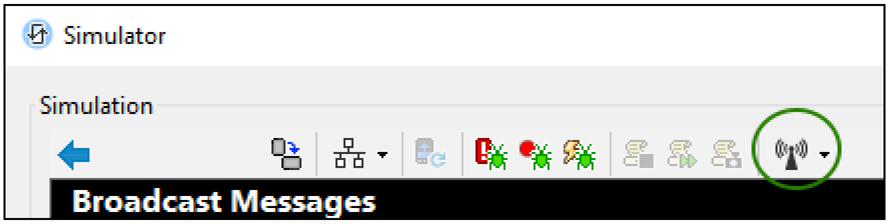
The action tree of `OnBroadcastReceive` is available at two levels:

- At the page level, in the `OnBroadcastReceive` page event. The action tree is accessed either via the Page Actions property[384] (in the Styles & Properties Pane[274]) or via the Page Actions dialog[400] (via the context menu of the page in the design).
- At the project level, via the Broadcast Actions project property[296].

You can define the actions to perform at either of these levels or at both levels. If an action tree has been defined at the page level for the active page, then this is executed. Otherwise, the action tree at the project level is executed—if such a tree exists. If no action has been defined at either level, then nothing happens when a broadcast message is received.

## Simulate actions on message received

You can simulate what happens when a Broadcast message is received. Press **F5** to start the simulator (*screenshot below*).



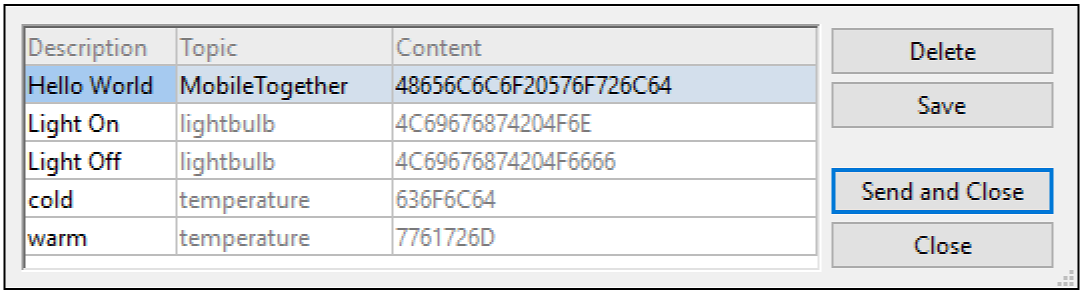
To simulate messages received, we use an **XML sample file** that is located by default in your MobileTogether Designer application folder. See the Options dialog (Simulation 2 tab)[1669] for the file path to this file. This file contains XML data that simulates messages received. So you can use it to test the action tree that will be executed when a message is received.

**Note:** If the XML sample file is in a write-protected folder (which is often the case with Windows application folders), you might need to open MobileTogether Designer with administrator privileges. This would enable you to save changes to the sample data (about which see below). To start MobileTogether Designer as an administrator, right-click its shortcut in Windows and select the command to start it as an administrator.

Start the MobileTogether Designer simulator by pressing **F5**. In the simulator[1355], use the *Simulate Reception of Message* toolbar button (*circled green in the screenshot above*) to open the Messages dialog.

In the dialog that appears, you can do the following:

- In the Messages dialog (*screenshot below*), the data displayed in the dialog's table is the data in the XML sample file. Each row represents one received message. The *Content* column contains the text of the message.

  | Description | Topic | Content |
  |---|---|---|
  | Hello World | MobileTogether | 48656C6C6F20576F726C64 |
  | Light On | lightbulb | 4C69676874204F6E |
  | Light Off | lightbulb | 4C69676874204F6666 |
  | cold | temperature | 636F6C64 |
  | warm | temperature | 7761726D |

  To simulate the reception of a message in the solution, select a message row and click **Send and Close**. The action tree of the solution will be executed with the data of the selected message and you can check whether actions were executed as you wanted. In the Messages dialog (*screenshot above*), you can edit the descriptions of messages and also delete a selected message. If you click **Save** after making changes, then the changes will be saved back to the XML sample file.

- The messages from the sample XML file are also available in the dropdown list of the *Simulate Reception of Message* toolbar button. Select a message in the dropdown list to simulate its receipt. Additionally, you can record a new message by first toggling on *Record a message* in the dropdown list of commands and then simulating the sending and receipt of a message. Whatever messages are sent while the recording is toggled on will be displayed in the Messages table (*screenshot above*). You can modify the descriptions as required and save the messages back to the XML sample file.

# 12.8 Barcode Scanners

A MobileTogether solution can be designed to read barcode data via a Zebra scanner, a Zebra mobile computer, or a Datalogic scanner and to then store and use data from the scan in the MobileTogether solution.

The following barcode scanners are supported.

| Scanner | Connection to MT Client | MT Client |
|---|---|---|
| Zebra | Bluetooth | Android, iOS |
| Zebra | USB | Android, Windows |
| Zebra mobile computer | Directly | Android |
| Datalogic | Directly | Android |

**Note:** Zebra mobile computer scanners and Datalogic scanners are Android mobile devices with integrated scanners. The MobileTogether client app is installed on these devices and, since it is on the device, it connects directly with the device's scanner.

## Overview: How barcode scanners work with MT solutions

The design mechanism for reading barcode data from the scanner and sending it to the solution and for using the data in the solution is as follows. The list below mixes design steps with runtime steps in order to better explain the design steps and their effects.

1. Start a connection with the scanner. In the design, there is an action for this 770 for each scanner. This kind of Connect action can be set on an event such as a button click. When you add a Connect action in the design, a page source 349 for the corresponding scanner type is added to the design's page sources 270.
2. In the design, you can configure the scanner's behavior and, optionally, specify that information about the scanner be sent to the page source together with the scanned barcode data. These settings are specified by adding, typically to the event of the previous step, Configure actions 770 for the relevant scanner type.
3. At runtime, after a connection to the scanner is established (by triggering a Connect action), the scanner can be used to read barcode data. Scanned data will be transmitted via the connection to the solution's scanner-related page source.
4. In the design, you can specify what action/s you want to carry out once the scanned data is received. These actions are defined on the **OnDataReceived** 296 event of the respective scanner—which are accessed via the project's Barcode Scanner Actions property 296.
5. In the design, after barcode scanning has been completed, end the connection to the scanner via the Disconnect action of the respective scanner 770.
6. After you have completed your design, you can simulate the use of barcode scanners in your workflow 1355.

## Connect to scanner

You can use the Connect action that corresponds to the barcode scanner you will be using. When a Connect action is added, a page source 349 for the corresponding scanner type is added to the design's page sources 270. The Connect actions for different scanner types are listed below together with their corresponding page sources. Click the links in the table for the descriptions of the different Connect actions. For a description of the page sources, see the corresponding section below.

| Connect action | Page source tree |
|---|---|
| Zebra Connect/Disconnect 772 | **$MT_ZEBRASCANNER** |
| Zebra Mobile Computer Connect/Disconnect 775 | **$MT_ZEBRAMOBILECOMPUTER** |
| Datalogic Connect/Disconnect 778 | **$MT_DATALOGICSCANNER** |

*Connecting to Zebra scanners via bluetooth*
When the Connect action to a Zebra scanner via Bluetooth is triggered at runtime, a dialog appears in the solution. It contains a list of available Bluetooth devices and a barcode. The end-user must select from the device list the Zebra scanner to use and then, with this scanner, scan the barcode in the solution's dialog. This starts the pairing of the MobileTogether client device with the Zebra scanner. You can use the `mt-zebra-scanner-connected()` 1262 function to check whether the Zebra scanner is connected.

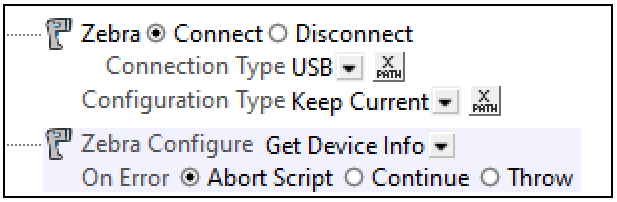
## Configure scanner

You can configure the properties of your scanner by using Configure actions for that scanner type. For example, on a Zebra scanner you can, among other options, (i) select whether to capture the barcode data or the barcode image and (ii) set the volume, duration, and tonal frequency of the scanner's beeper. Since each Configure action sets one property, you can add as many Configure actions as you want.

Click the links below to see the descriptions of the different Configure actions.

- Zebra Configure 773
- Zebra Mobile Computer Configure 777
- Datalogic Configure 779

*Get scanner information*
You can set the Config action of any scanner type to also send information about the scanner (such as its serial number or version number) to the solution when a barcode is scanned. Do this by adding, in the design, a Configure action that is set to *Get Device Info (see screenshot of Zebra Configure below)*.

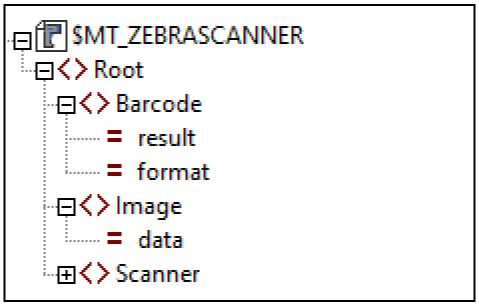


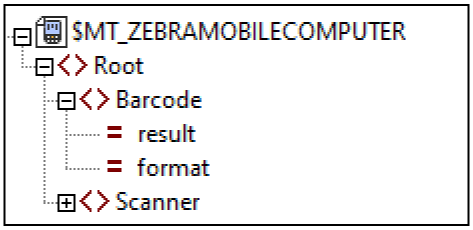
Zebra ● Connect ○ Disconnect
Connection Type USB
Configuration Type Keep Current
Zebra Configure Get Device Info
On Error ● Abort Script ○ Continue ○ Throw

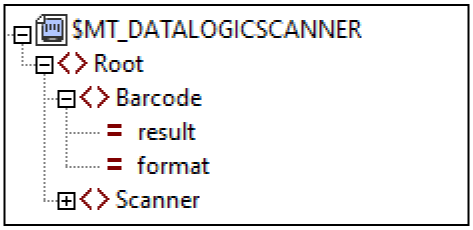
The scanner information that is sent varies according to the scanner type and is stored in the respective page source's **`/Root/Scanner`** node (*see next section below*).

## Page source trees for barcode-scan data

The page source tree corresponding to a particular type of scanner is added to the design whenever a Connect action for that scanner type (*see above*) is added: **$MT_ZEBRASCANNER**, **$MT_ZEBRAMOBILECOMPUTER**, **$MT_DATALOGICSCANNER**. The page source is removed when the Connect action is deleted.







When a barcode is scanned at runtime, barcode data (and, if configured, scanner data) is sent to the page source. Each page source has a common structure.

The **`Root`** node of each contains two child elements: `Barcode` and `Scanner`, which contain, respectively, data about the barcode and the scanner. The barcode data is stored in `Barcode/@result` and the barcode type (EAN, ISBN, etc) is stored in `Barcode/@format`.

On scanning a barcode at runtime, scanner information (such as its serial number) is sent to a page source only if a scanner Configure action has been set to *Get Device Info*.

The **`Root`** node of the **`$MT_ZEBRASCANNER`** page source has an additional child element—named `Image`. If a Zebra scanner's Configure action has its *Set Scanner Mode* option set to *Image*, then the barcode image is scanned and stored in Base64 format in the page source's `Image/@data` attribute.

**Note:** Data is left in these page sources till (i) the data is overwritten by data from a new scan or (ii) the solution is closed, in which case the data is deleted.

## Actions for barcode-scanner events

For each scanner type, you can specify actions to perform when data is received in the tree following a barcode scan:

- Zebra Scanner OnDataReceived
- Zebra Mobile Computer OnDataReceived
- Datalogic Scanner OnDataReceived

To define these actions, click the **Additional Options** button of the project property Barcode Scanner Actions 296. The **`OnDataReceived`** actions will be executed in sequence immediately the scan data is received in the page source.

For the Zebra scanner, which needs to have a Bluetooth or USB connection with the solution, actions can be defined for two other important events:

- Zebra Scanner OnConnectionEstablished

- Zebra Scanner OnConnectionTerminated

## Disconnect from scanner

After barcode scanning for the solution has been completed, it is best to disconnect the scanner form the solution. This way, you will avoid inadvertently scanned data being passed and stored in the solution and you will release the scanner for other tasks not related to the solution. To disconnect a scanner, use the respective Disconnect action:

- Zebra Connect/Disconnect[772]
- Zebra Mobile Computer Connect/Disconnect[775]
- Datalogic Connect/Disconnect[778]

## Simulations

Simulations[1355] now offer the possibility to simulate barcode scans with the supported scanners. Various aspects of the runtime scenario, such as a Bluetooth connection, establishing connections to the scanner, and receiving data can be simulated directly from the simulator. Default settings for some of these simulator options can be set in the Simulation 1 tab[1668] of the Options tab.

# 12.9 Charts

Charts provide a graphical representation of data in the source document. A chart is set up by defining XPath expressions to specify a sequence of items for each axis of the chart. MobileTogether Designer then automatically generates the chart. The table below shows the types of charts that can be created, and the kind of data items that are required for each of the chart's axes.

| Chart type | X-Axis (Category) | Y-Axis (Value) | Number of Series (on Z-Axis) |
|---|---|---|---|
| Pie Charts (2D, 3D) | Text | Numeric | 1 |
| Bar Charts Ungrouped (2D, 3D) | Text | Numeric | 1 |
| Bar Charts Grouped (2D, 3D) | Text | Numeric | > 1 |
| Category Line Graphs | Text | Numeric | 1 line = 1 series |
| Value Line Graphs | Numeric | Numeric | 1 line = 1 series |
| Area and Stacked Area Charts | Text | Numeric | 1 area = 1 series |
| Candlestick Charts | Text | Numeric | 3 or 4 |
| Gauge | — | Numeric | 1 |
| Overlay Charts | Text | Numeric | = 1 or > 1 per chart |

This section is organized as follows:

- Chart Data Selection 1152 explains how to select data for the different axes
- Chart Settings and Appearance 1171 describes how to define the properties of charts

## 12.9.1 Creating and Configuring Charts

*This section*:

### Creating a chart

To insert a chart in the design, drag the Chart control 438 from the Controls Pane 266 to the location in the design where you wish to insert the chart.

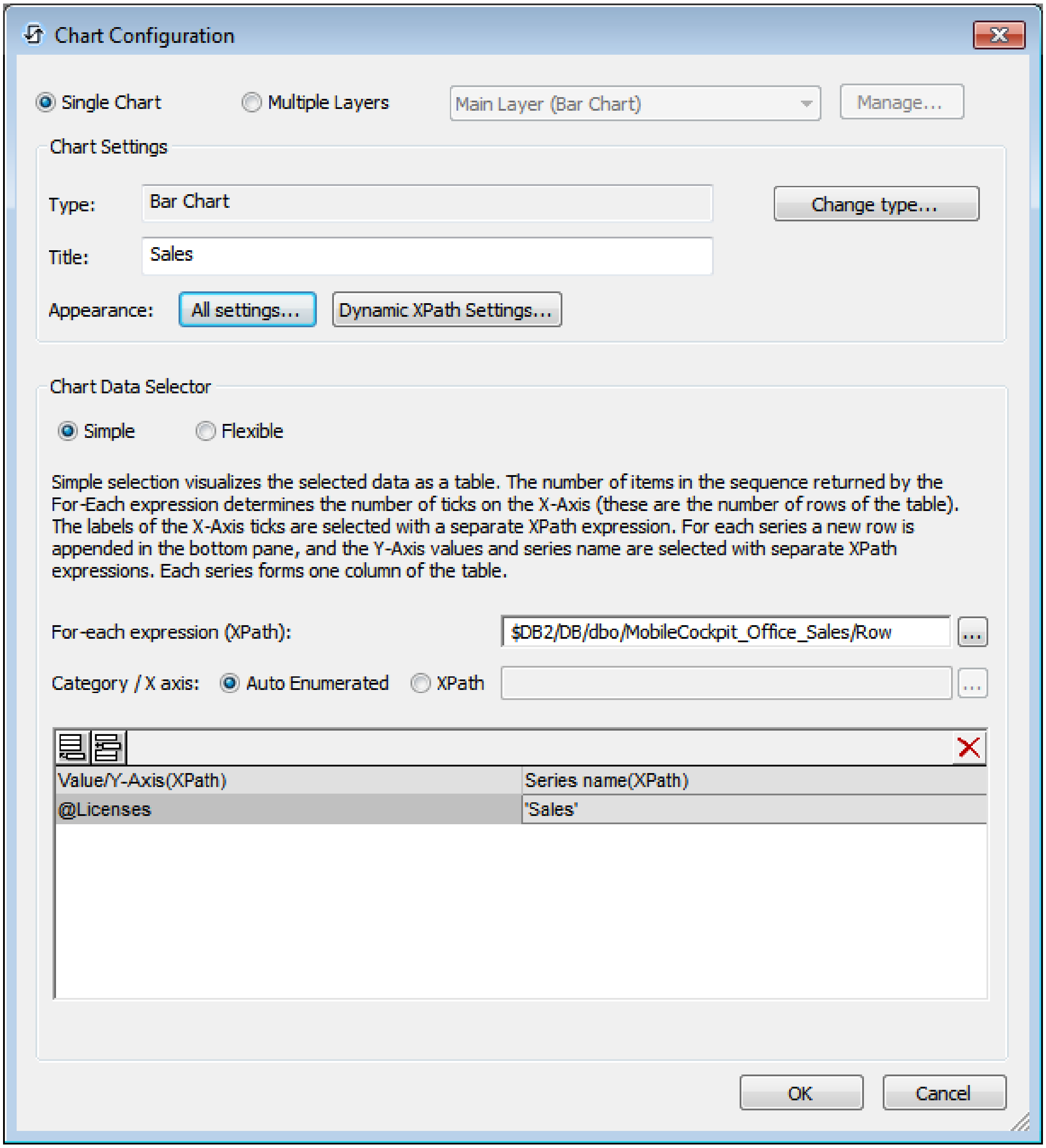
## The context node

Drag an XML node from the Page Sources Pane 270 onto the chart in the design to make this XML node the context node of the chart's XPath expressions. You can change the chart's context node at any time by dragging a new XML node onto the chart. It is important to be aware of the chart's context node because this context node is the starting point of path locators in XPath expressions.

## The Chart Configuration dialog

A chart can be configured in the Chart Configuration dialog (*screenshot below*) at the time that the chart is created. Configuring a chart involves: (i) specifying data selection for the charts's axes 1152, and (ii) defining the chart's properties 1171. These settings can be edited at any later time. After assigning an XML context node to the chart, do either of the following to bring up the Chart Configuration dialog:

- Double-click the chart
- Select the chart in the design, then click the Edit XPath button of the *Chart Settings* property in the Styles & Properties Pane



● Single Chart ○ Multiple Layers

Type: Bar Chart

Title: Sales

● Simple ○ Flexible

Simple selection visualizes the selected data as a table. The number of items in the sequence returned by the For-Each expression determines the number of ticks on the X-Axis (these are the number of rows of the table). The labels of the X-Axis ticks are selected with a separate XPath expression. For each series a new row is appended in the bottom pane, and the Y-Axis values and series name are selected with separate XPath expressions. Each series forms one column of the table.

For-each expression (XPath): $DB2/DB/dbo/MobileCockpit_Office_Sales/Row

Category / X axis: ● Auto Enumerated ○ XPath

| Value/Y-Axis(XPath) | Series name(XPath) |
|---|---|
| @Licenses | 'Sales' |

The Chart Configuration dialog has three parts:

- *Single or Multiple layers*[1168]: Multiple layers can be selected to create overlay charts; charts are overlaid one upon the other to produce a composite
- *Chart Settings*[1171]: To select the chart type and define the appearance of the chart
- *Chart Data Selection*[1152]: To select the data for the various axes of the chart using either the simple option[1156] or flexible option[1161]

### Editing chart settings and data selection

If you wish to change chart settings or a chart's data selection after a chart has been created, right-click the chart in the design and select **Chart Creation Settings**. The Chart Configuration dialog (*screenshot above*) of that chart appears. You can edit the chart's settings or data selection in the dialog, then click **OK** to finish.

## 12.9.2 Chart Data Selection

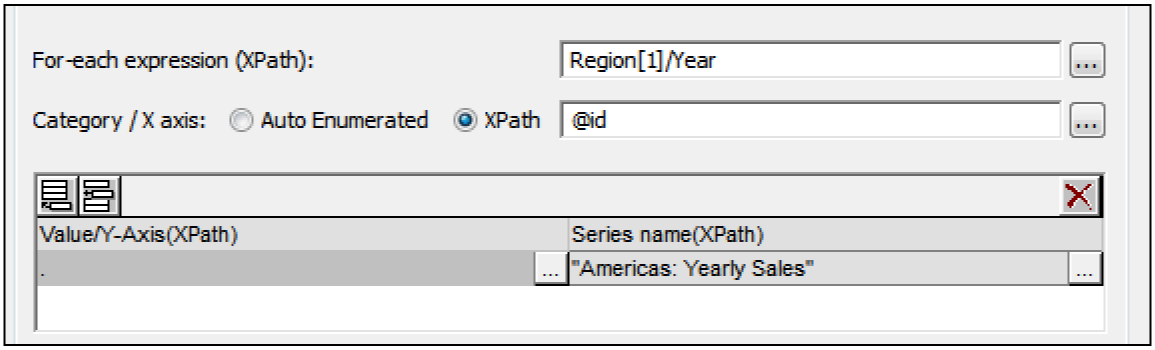
This topic contains simple examples to illustrate how chart data selection works.

⊟ XML file used in chart examples: YearlySales.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <Region id="Americas">
      <Year id="2005">30000</Year>
      <Year id="2006">90000</Year>
      <Year id="2007">120000</Year>
      <Year id="2008">180000</Year>
      <Year id="2009">140000</Year>
      <Year id="2010">100000</Year>
   </Region>
   <Region id="Europe">
      <Year id="2005">50000</Year>
      <Year id="2006">60000</Year>
      <Year id="2007">80000</Year>
      <Year id="2008">100000</Year>
      <Year id="2009">95000</Year>
      <Year id="2010">80000</Year>
   </Region>
   <Region id="Asia">
      <Year id="2005">10000</Year>
      <Year id="2006">25000</Year>
      <Year id="2007">70000</Year>
      <Year id="2008">110000</Year>
      <Year id="2009">125000</Year>
      <Year id="2010">150000</Year>
   </Region>
</Data>
```

## Chart data selection with four XPath expressions

The screenshot below shows the Chart Configuration dialog[1150], at the bottom of which is the Chart Data Selector pane with fields for entering the four XPath expressions for data selection.



For-each expression (XPath): Region[1]/Year

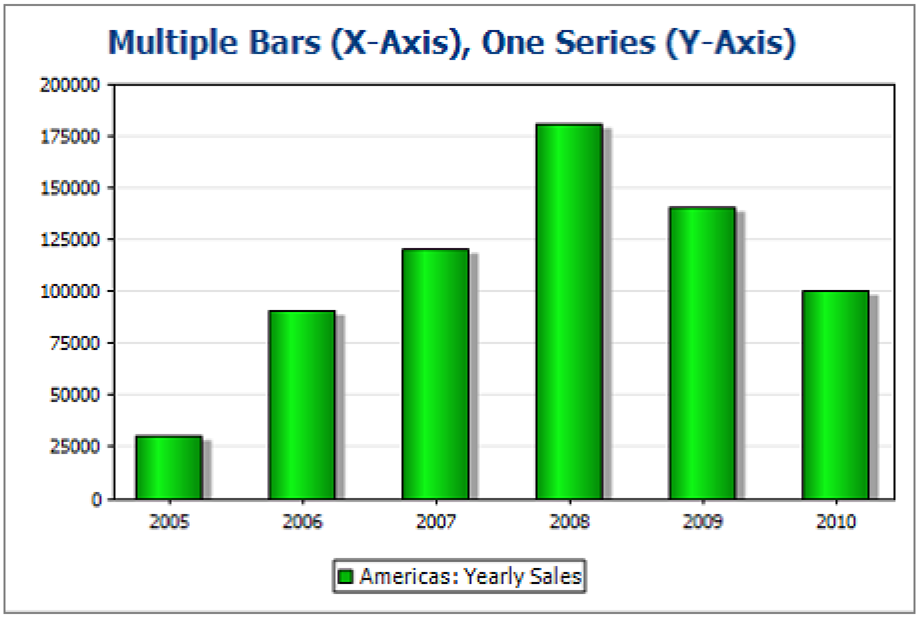
Category / X axis: ○ Auto Enumerated ● XPath @id

| Value/Y-Axis(XPath) | Series name(XPath) |
|---|---|
| . | "Americas: Yearly Sales" |

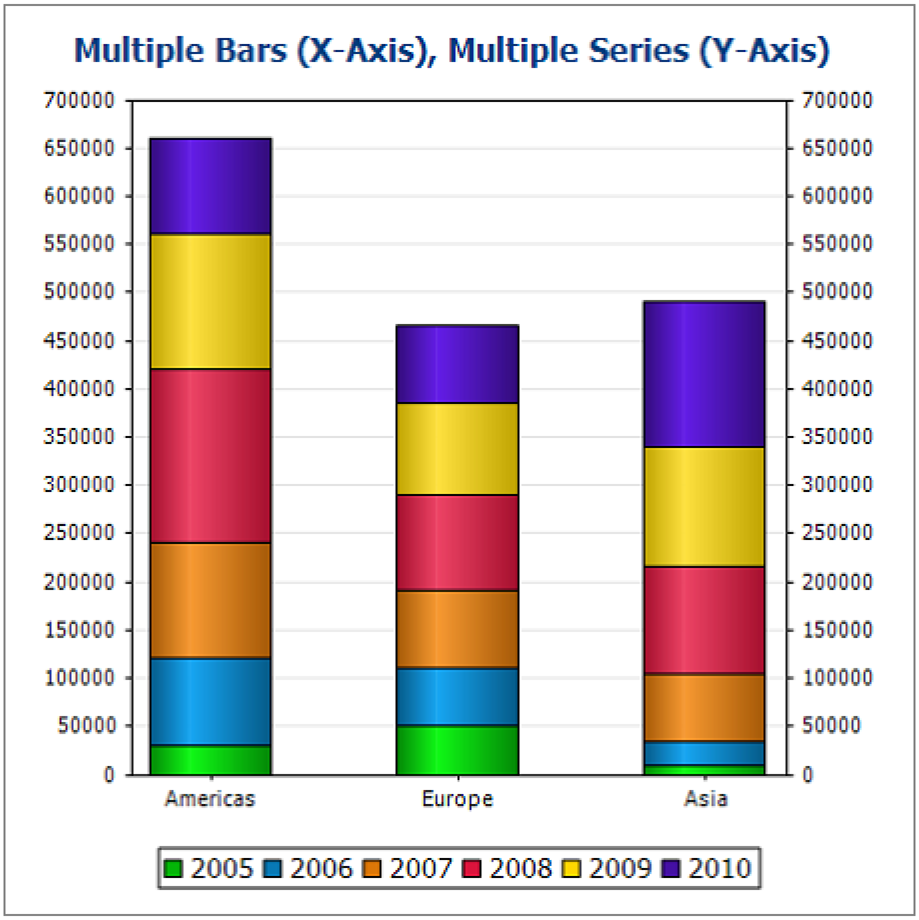
The four XPath expressions in the Chart Data Selector pane work together and do the following:

| XPath | Description |
|---|---|
| *For-Each* | • Sets the context for the other three XPath expressions<br>• Sets the number of items in the returned sequence as the number of ticks on the X-Axis.<br>• In the case of the screenshot above, the `Region[1]/Year` expression returns six node items; so there will be six ticks on the X-Axis (*see screenshot below*). |
| *X-Axis* | • The items in the returned sequence provide the label text for the corresponding ticks on the X-Axis.<br>• In the example shown above, the `@id` expression returns the `id` attribute value of each `Year` element. These values become the labels of the corresponding ticks (*see screenshot below*).<br>• Since we have specified that this will be a bar chart, bars are drawn at the ticks. |
| *Y-Axis* | • The Y-Axis can display multiple series, each of which is defined in one row of the Y-Axis table.<br>• Each series is defined by two XPath expressions: one for the series value, the other for the series name.<br>• In our example, the `self::node()` XPath expression (indicated by its abbreviated form of a period) selects the current node, which is the `Year` element that is the context node. So, for each `Year` element (represented by a bar on the X-Axis), the `Year` element's content will be read as the Y-Axis value of that year, and therefore plotted as the height of the bar (*see screenshot below*). The screenshot further below shows a chart with multiple series on the Y-Axis. |
| *Series name* | • This expression provides the legend text for the series. In our example, the legend text (which appears at the bottom of the chart, *screenshot below*) is obtained from an XPath expression that is a text string (*see screenshot above*). |

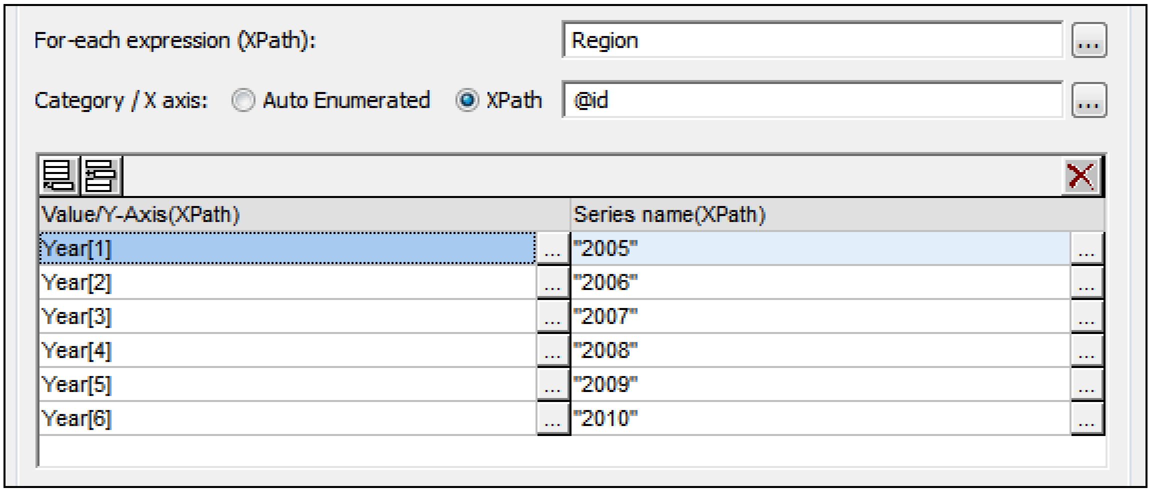
A bar chart that is generated for the data selection shown in the screenshot above and the XML data in `YearlySales.xml`[1152] looks like the chart in the screenshot below.



The screenshot above shows a bar chart with a single series, while that below is of a stacked bar chart with multiple series. In the latter example, the value of each series is stacked on to the bar.



The XPath expressions of this chart are shown in the screenshot below.



For-each expression (XPath): Region

Category / X axis: ○ Auto Enumerated ● XPath @id

| Value/Y-Axis(XPath) | Series name(XPath) |
|---|---|
| Year[1] | "2005" |
| Year[2] | "2006" |
| Year[3] | "2007" |
| Year[4] | "2008" |
| Year[5] | "2009" |
| Year[6] | "2010" |

**Note:** Pie charts and gauge charts have a single nominal series, which requires no name. So if a series name is entered in the data selection, it is ignored. For ungrouped bar charts, however, the name of the single series, if present, is used for the legend. For gauge charts, in addition to any Series Name entry being ignored, the X-Axis data selection is also ignored; only the Y-Axis selection is used for gauge charts.

## 12.9.2.1 Chart Data Selection: Simple

*This section*:

### Introduction

In the Chart Data Selector pane of the Chart Configuration dialog 1150, the Simple option enables the data selection to be visualized as a table. We use the XML document that is listed below to explain the visualization.

⊟ XML file used in chart examples: YearlySales.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <Region id="Americas">
      <Year id="2005">30000</Year>
      <Year id="2006">90000</Year>
      <Year id="2007">120000</Year>
      <Year id="2008">180000</Year>
      <Year id="2009">140000</Year>
      <Year id="2010">100000</Year>
   </Region>
   <Region id="Europe">
      <Year id="2005">50000</Year>
      <Year id="2006">60000</Year>
      <Year id="2007">80000</Year>
      <Year id="2008">100000</Year>
      <Year id="2009">95000</Year>
      <Year id="2010">80000</Year>
   </Region>
   <Region id="Asia">
      <Year id="2005">10000</Year>
      <Year id="2006">25000</Year>
      <Year id="2007">70000</Year>
      <Year id="2008">110000</Year>
      <Year id="2009">125000</Year>
      <Year id="2010">150000</Year>
   </Region>
</Data>
```

## The context node

In the design, drag an XML node from the Page Sources Pane 270 to make this node the context node of the chart's XPath expressions. You can change the chart's context node by dragging a new XML node onto the chart. It is important to be aware of the chart's context node, since this context node is the starting point of path locators in XPath expressions.

## Selecting data for the X and Y axes

In the Chart Data Selector pane (*screenshot below*) we make the data selection as shown in the screenshot. Since the chart has been inserted within the `Data` node, the context node for the For-Each expression is the `Data` node.

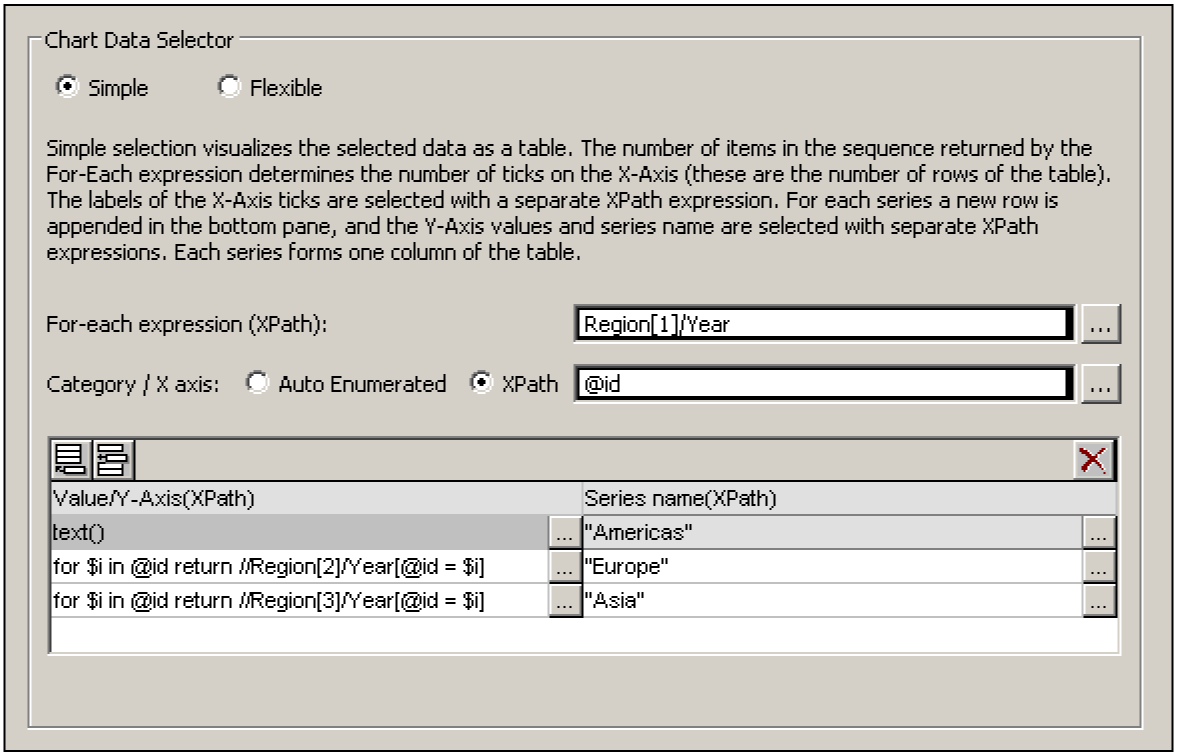


Chart Data Selector

● Simple ○ Flexible

Simple selection visualizes the selected data as a table. The number of items in the sequence returned by the For-Each expression determines the number of ticks on the X-Axis (these are the number of rows of the table). The labels of the X-Axis ticks are selected with a separate XPath expression. For each series a new row is appended in the bottom pane, and the Y-Axis values and series name are selected with separate XPath expressions. Each series forms one column of the table.

For-each expression (XPath): Region[1]/Year

Category / X axis: ○ Auto Enumerated ● XPath @id

| Value/Y-Axis(XPath) | Series name(XPath) |
|---|---|
| text() | "Americas" |
| for $i in @id return //Region[2]/Year[@id = $i] | "Europe" |
| for $i in @id return //Region[3]/Year[@id = $i] | "Asia" |

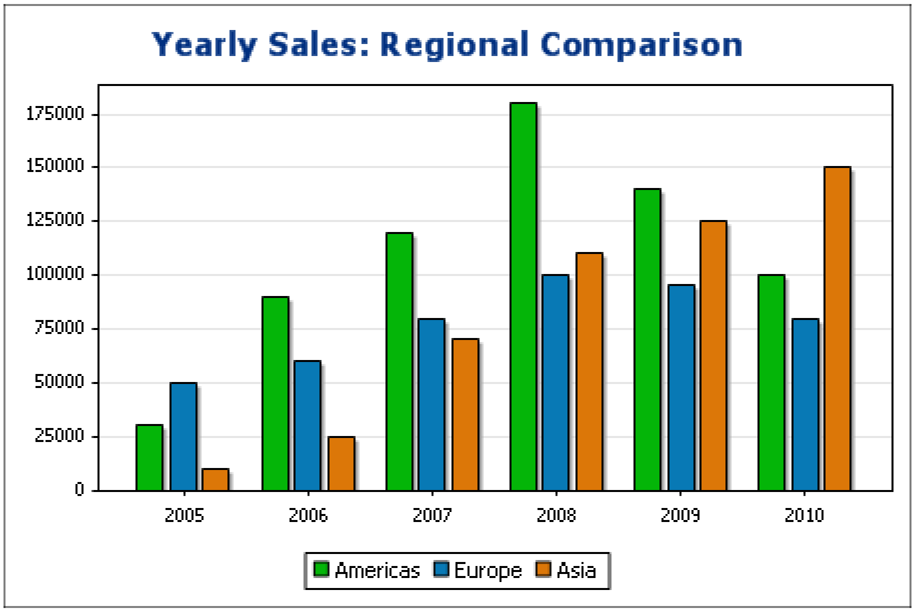
The chart data table can be visualized as the table below. What happens is that for each `Region[1]/Year` element a row is created and the X-Axis and Y-Axis XPath expressions are evaluated within the respective `Region[1]/Year` element's context.

| For-Each XPath | X-Axis | Y-Axis for Series | | |
|---|---|---|---|---|
| | | Americas | Europe | Asia |
| `Region[1]/Year[1]` | `@id` | `text()` | `XPath-1` | `XPath-2` |
| `Region[1]/Year[2]` | `@id` | `text()` | `XPath-1` | `XPath-2` |

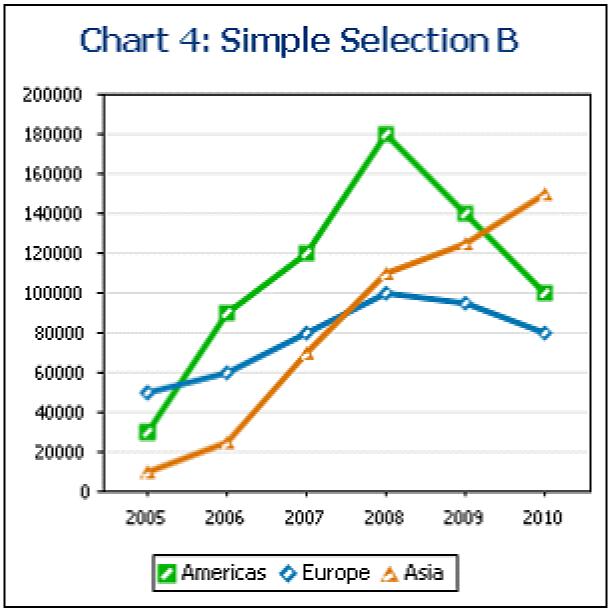
| | | | | |
|---|---|---|---|---|
| `Region[1]/Year[3]` | `@id` | `text()` | `XPath-1` | `XPath-2` |
| `Region[1]/Year[4]` | `@id` | `text()` | `XPath-1` | `XPath-2` |
| `Region[1]/Year[5]` | `@id` | `text()` | `XPath-1` | `XPath-2` |
| `Region[1]/Year[6]` | `@id` | `text()` | `XPath-1` | `XPath-2` |

- The For-Each expression `Region[1]/Year` returns six nodes (which become the rows of the table). The number of items in the sequence returned by the For-Each expression determines the **number of ticks** on the X-Axis.
- The XPath expression for the X-Axis returns the `@id` attribute value of each `Region[1]/Year` element. These values will be the **labels of the X-Axis ticks**. If there are more labels than ticks then extra ticks will be generated so that all labels are plotted. If there are fewer labels than ticks, then the latter ticks (for which no corresponding labels exist) will be unlabeled. The Auto-Enumerated option generates a sequence of integers starting with 1, and assigns each integer sequentially to an X-Axis tick.
- The XPath expression for the `Americas` series (`text()`) returns the content of each of the `Region[1]/Year` elements. This expression could also have been one similar to that for the Europe and Asia series (explained below)—as long as it efficiently returns the values we want.
- The XPath expression for the `Europe` series is: `for $i in @id return //Region[2]/Year[@id=$i]`. This expression does the following: (i) looks for the current `Region[1]/Year/@id` attribute value, (ii) returns the content of the `Region[2]/Year` element that has the same `@id` value as the `@id` value of the current `Region[1]/Year` element.
- The XPath expression for the `Asia` series works in a similar way to the XPath expression for the `Europe` series.

The bar chart generated with this data selection would look something like this:



The line graph chart for this data selection would look like this:



### If the For-Each expression returns items that are not nodes

Since the number of X-Axis ticks is primarily dependent on the number of items returned by the For-Each XPath expression, the XPath expression in the screenshot below (`distinct-values(//Year/@id)`), which returns the six unique year values, will also generate six ticks on the X-Axis. The items returned by the sequence, however, are atomic values, not nodes. Consequently, although they can be used as context items, they cannot be used as context nodes for locating nodes in the XML tree. They can, however, be used to locate nodes on the basis of the equality of values—which is how we will use them.

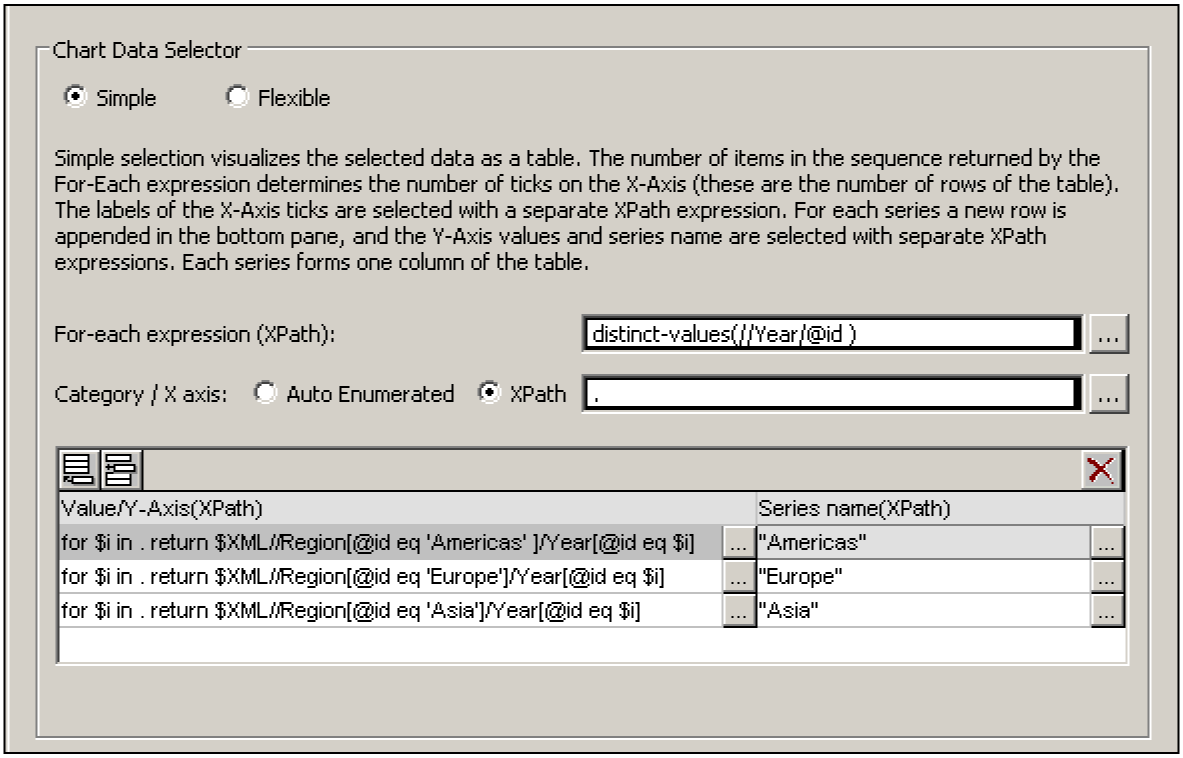


Chart Data Selector

● Simple ○ Flexible

Simple selection visualizes the selected data as a table. The number of items in the sequence returned by the For-Each expression determines the number of ticks on the X-Axis (these are the number of rows of the table). The labels of the X-Axis ticks are selected with a separate XPath expression. For each series a new row is appended in the bottom pane, and the Y-Axis values and series name are selected with separate XPath expressions. Each series forms one column of the table.

For-each expression (XPath): distinct-values(//Year/@id )

Category / X axis: ○ Auto Enumerated ● XPath .

| Value/Y-Axis(XPath) | Series name(XPath) |
|---|---|
| for $i in . return $XML//Region[@id eq 'Americas' ]/Year[@id eq $i] | "Americas" |
| for $i in . return $XML//Region[@id eq 'Europe']/Year[@id eq $i] | "Europe" |
| for $i in . return $XML//Region[@id eq 'Asia']/Year[@id eq $i] | "Asia" |

In the data selection shown in the screenshot above, note the following:

- The X-Axis and Y-Axis data selections use the atomic values returned by the For-Each expression, respectively, as direct output and as filter test values.
- Location steps in XPath expressions start at the document node (the `$XML` in `$XML//Region...`). This is necessary because the atomic values provide no locational context.

The chart data table would evaluate to the following:

| For-Each XPath | X-Axis | Y-Axis for Series | | |
|---|---|---|---|---|
| | | Americas | Europe | Asia |
| 2005 | 2005 | XPath-1 | XPath-2 | XPath-3 |
| 2006 | 2006 | XPath-1 | XPath-2 | XPath-3 |
| 2007 | 2007 | XPath-1 | XPath-2 | XPath-3 |
| 2008 | 2008 | XPath-1 | XPath-2 | XPath-3 |
| 2009 | 2009 | XPath-1 | XPath-2 | XPath-3 |
| 2010 | 2010 | XPath-1 | XPath-2 | XPath-3 |

## 12.9.2.2 Chart Data Selection: Flexible

*This section*:

### About flexible chart data selection

In the Chart Data Selector pane (*screenshot below*) of the Chart Configuration dialog 1150, the Flexible option enables the Series Axis (Z-Axis), X-Axis, and Y-Axis data to be selected freely using XPath expressions. The XPath expression for an axis returns the sequence of items that are to be plotted on that axis. These sequences (of items) for the axes are then collated to generate the chart.

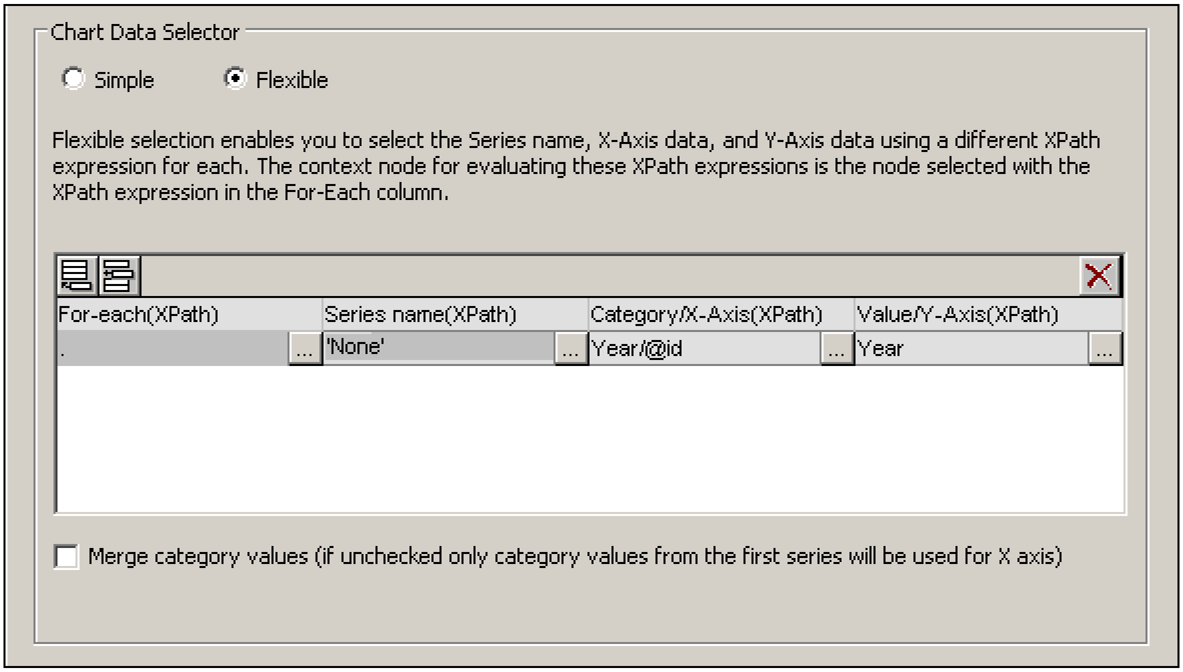


Chart Data Selector

○ Simple ● Flexible

Flexible selection enables you to select the Series name, X-Axis data, and Y-Axis data using a different XPath expression for each. The context node for evaluating these XPath expressions is the node selected with the XPath expression in the For-Each column.

| For-each(XPath) | Series name(XPath) | Category/X-Axis(XPath) | Value/Y-Axis(XPath) |
|---|---|---|---|
| . | 'None' | Year/@id | Year |

□ Merge category values (if unchecked only category values from the first series will be used for X axis)

Note the following points:

- A series refers to a series of values plotted for a set of X-Axis (Category Axis) ticks. A second series would plot a second set of values on the same X-Axis ticks. For example, if the X-Axis represented the years 2008, 2009, and 2010, and the Y-Axis represented sales turnover, then Series 1 could represent America (sales in America for those three years) while Series 2 could represent Europe (sales in Europe for those three years). If this data were selected for a bar chart, then for each year (2008, 2009, 2010) on the X-Axis, there would be two bars (America and Europe), one for each series. In the case of pie charts and single-bar charts, only one series is possible. See the chart type table 1171 for more information about each chart type.
- Each row in the Chart Data Selector pane represents a series.

- The chart's XPath context node is defined by dropping a node from the Page Source Pane onto the chart control in the design.
- The XPath expression in the For-Each column provides the context for the evaluation of each of the other three XPath expressions. The For-Each XPath expression is itself evaluated in the context of the node in the design within which it was inserted.

The following examples illustrate important points to consider when selecting data for the axes. They reference the XML document listed below.

⊟ XML file used in chart examples: YearlySales.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Data xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="YearlySales.xsd">
   <Region id="Americas">
      <Year id="2005">30000</Year>
      <Year id="2006">90000</Year>
      <Year id="2007">120000</Year>
      <Year id="2008">180000</Year>
      <Year id="2009">140000</Year>
      <Year id="2010">100000</Year>
   </Region>
   <Region id="Europe">
      <Year id="2005">50000</Year>
      <Year id="2006">60000</Year>
      <Year id="2007">80000</Year>
      <Year id="2008">100000</Year>
      <Year id="2009">95000</Year>
      <Year id="2010">80000</Year>
   </Region>
   <Region id="Asia">
      <Year id="2005">10000</Year>
      <Year id="2006">25000</Year>
      <Year id="2007">70000</Year>
      <Year id="2008">110000</Year>
      <Year id="2009">125000</Year>
      <Year id="2010">150000</Year>
   </Region>
</Data>
```

## One row, one series

Say we wish to generate a 2D bar chart for each `Region` element (there are three such elements: for the Americas, Europe, and Asia). Let us create the chart in the design by dropping the chart control at the desired location in the design. We create the `Region` element node as the chart's XPath context node by dropping it onto the chart control. The context node of the For-Each XPath expressions in the Chart Data Selector will therefore be the `Region` element.

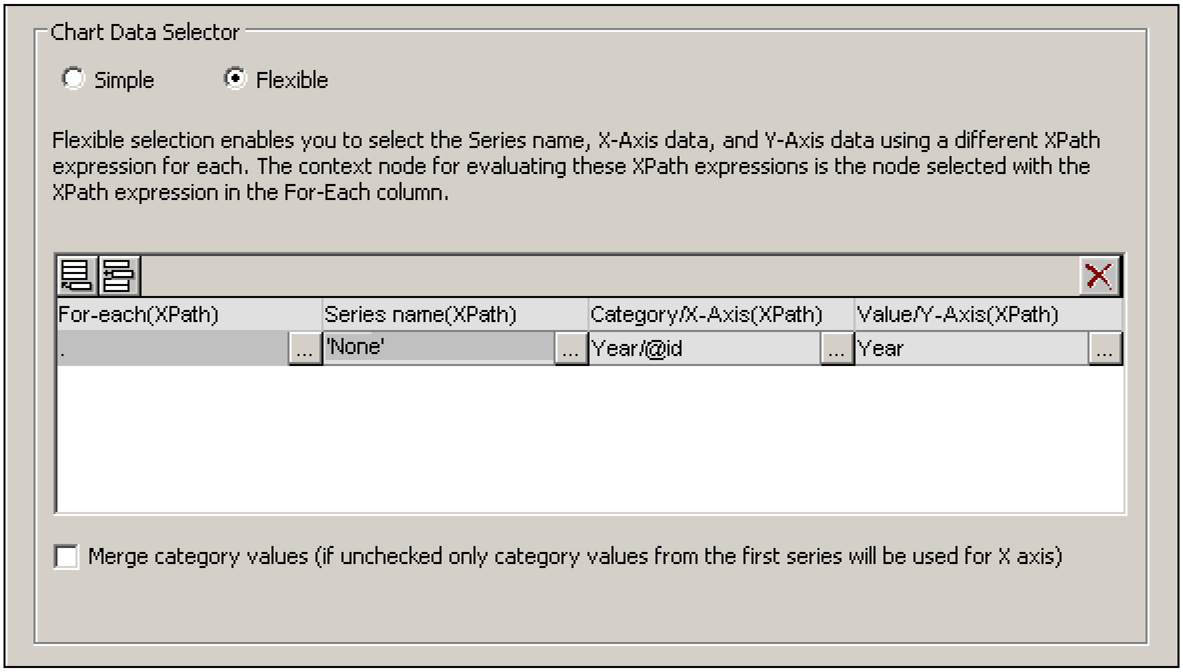


Chart Data Selector

○ Simple ● Flexible

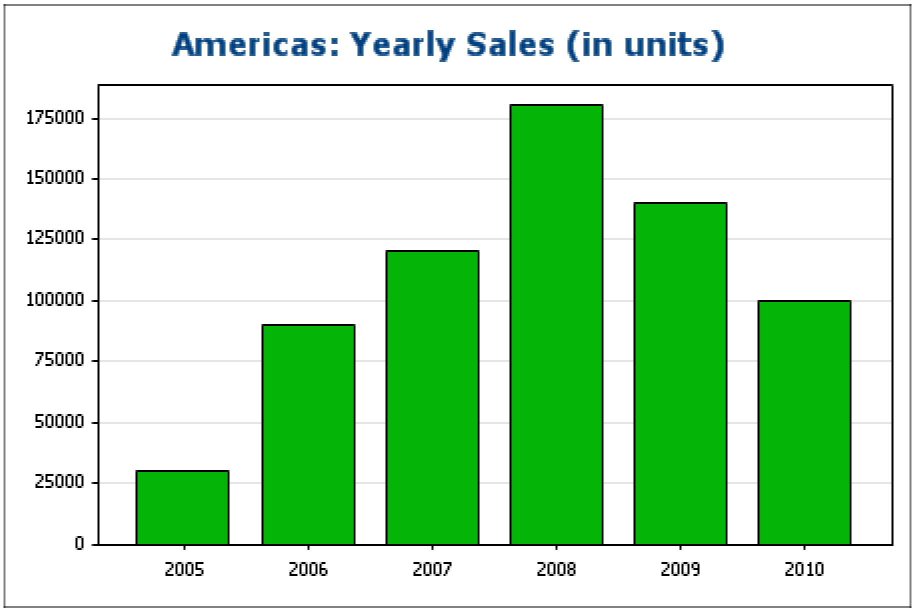
Flexible selection enables you to select the Series name, X-Axis data, and Y-Axis data using a different XPath expression for each. The context node for evaluating these XPath expressions is the node selected with the XPath expression in the For-Each column.

| For-each(XPath) | Series name(XPath) | Category/X-Axis(XPath) | Value/Y-Axis(XPath) |
|---|---|---|---|
| . | 'None' | Year/@id | Year |

□ Merge category values (if unchecked only category values from the first series will be used for X axis)

In the chart data selection shown in the screenshot above, the For-Each expression returns the current node (which is the `Region` element), so the `Region` element will be the context node for all the other three XPath expressions (series, X-Axis, and Y-Axis). Since there is only one series in this chart, we do not need a series name and so we leave this column blank. The X-Axis selection returns six values. Six will therefore be the number of ticks on the X-Axis and the six items of the sequence will be the respective labels of the X-Axis ticks. The Y-Axis selection also returns six items, each of which is plotted on the Y-Axis for its corresponding X-Axis tick. Since the chart was created within the `Region` element, a chart will be created for each of the three `Region` elements. For each chart, that particular `Region` element's descendant nodes will be used.

The chart for the `Americas` region would look something like this in the output:



## Three rows, three series, category values not merged

To create multiple series, additional rows can be added to the chart data selection, as shown in the screenshot below.

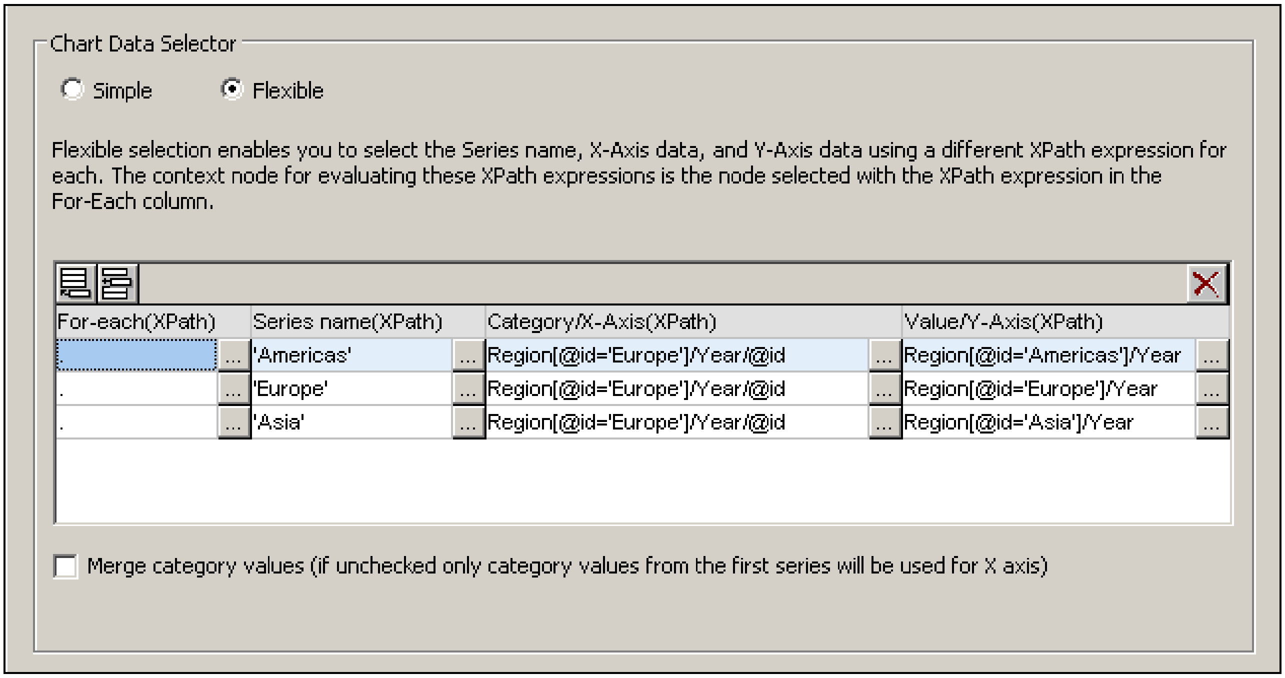


Chart Data Selector

○ Simple ● Flexible

Flexible selection enables you to select the Series name, X-Axis data, and Y-Axis data using a different XPath expression for each. The context node for evaluating these XPath expressions is the node selected with the XPath expression in the For-Each column.

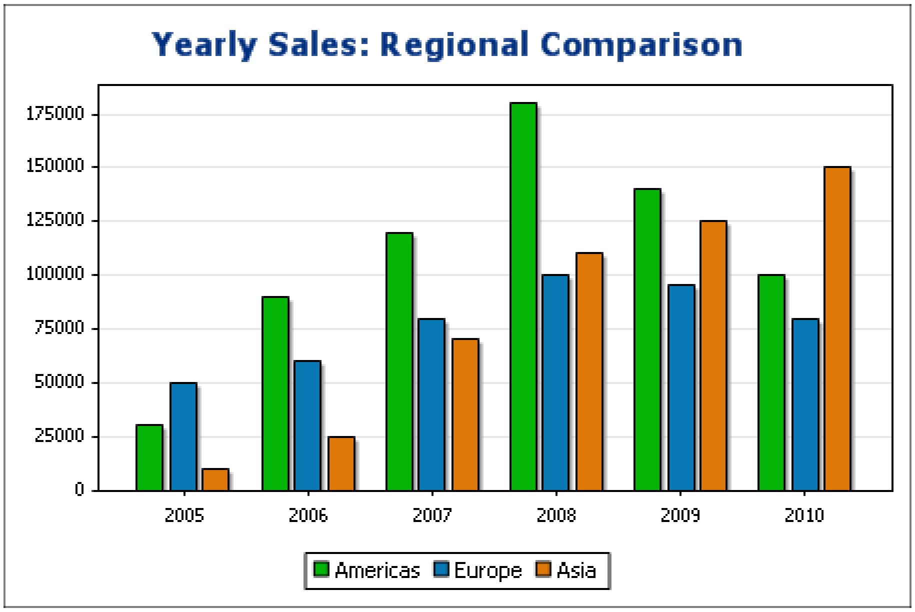
| For-each(XPath) | Series name(XPath) | Category/X-Axis(XPath) | Value/Y-Axis(XPath) |
|---|---|---|---|
| . | 'Americas' | Region[@id='Europe']/Year/@id | Region[@id='Americas']/Year |
| . | 'Europe' | Region[@id='Europe']/Year/@id | Region[@id='Europe']/Year |
| . | 'Asia' | Region[@id='Europe']/Year/@id | Region[@id='Asia']/Year |

□ Merge category values (if unchecked only category values from the first series will be used for X axis)

The important points to note about the data selection above are:

- Each row defines a series and all rows have the `Data` element as its context node (since the chart has the `Data` node as its XPath context node).
- The first row is set to define the Americas series and is given a string expression as its series name. The X-Axis values are selected using the `Year/@id` values of the Europe region (it doesn't matter which region is selected since all have the same `Year/@id` values). The Y-Axis values of the first (Americas) series are selected for the Americas region using a predicate filter.
- The second and third series follow the same pattern as the first series. Note, however, that the X-Axis selection for each series is identical. But since the *Merge Category Values* check box is not checked, the second and third expressions are ignored. (Even if the values were merged, it would not make a difference because the values of each series are identical; only new distinct values would be added to the category values.)

The chart generated with the data selection above would look something like this:



## Three rows, three series, category values merged

The data selection in this example (*see screenshot below*) is different from the previous example in three respects: (i) the X-Axis selection for the third series has an extra item (`2011`) added to the series, and (ii) the *Merge Category Values* check box has been checked, and (iii) the Y-Axis tick interval has been manually set 1193 to `20000`.

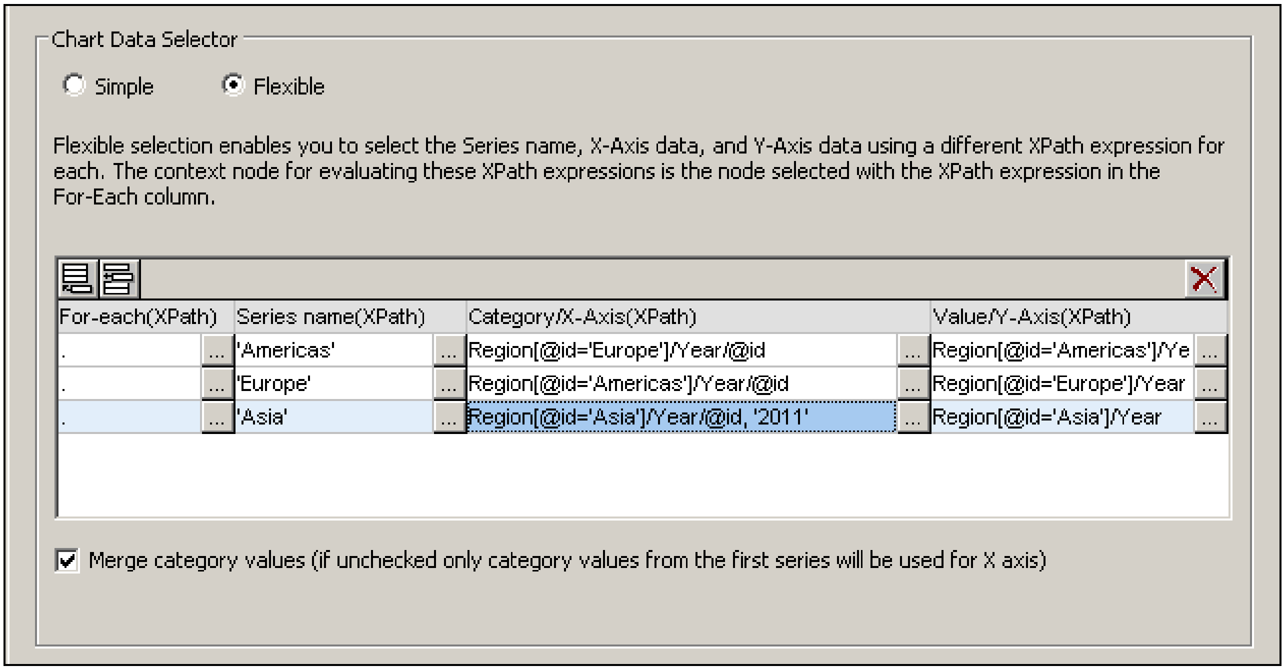


Chart Data Selector

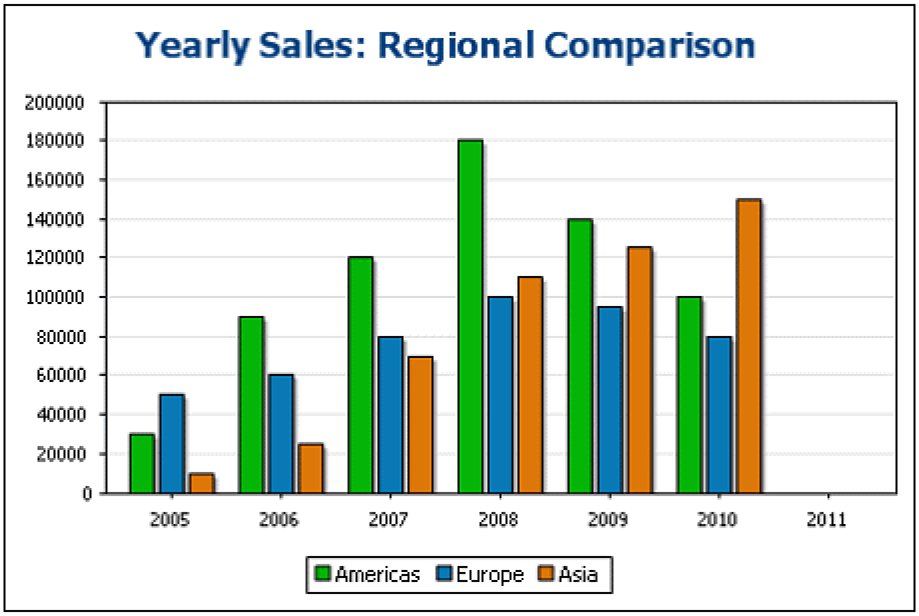
○ Simple ● Flexible

Flexible selection enables you to select the Series name, X-Axis data, and Y-Axis data using a different XPath expression for each. The context node for evaluating these XPath expressions is the node selected with the XPath expression in the For-Each column.

| For-each(XPath) | Series name(XPath) | Category/X-Axis(XPath) | Value/Y-Axis(XPath) |
|---|---|---|---|
| . | 'Americas' | Region[@id='Europe']/Year/@id | Region[@id='Americas']/Ye |
| . | 'Europe' | Region[@id='Americas']/Year/@id | Region[@id='Europe']/Year |
| . | 'Asia' | Region[@id='Asia']/Year/@id, '2011' | Region[@id='Asia']/Year |

☑ Merge category values (if unchecked only category values from the first series will be used for X axis)

The effect of this change is to add one new item (`2011`) to the X-Axis result sequence. The chart would look something like this:



## One row, three series

The chart in this example has the `Data` node (see XML document above) as its XPath context node. Only one row is used for data selection, but it generates three series. This is because the XPath expression in the For-Each column returns a sequence of three items, thus implicitly creating three series.

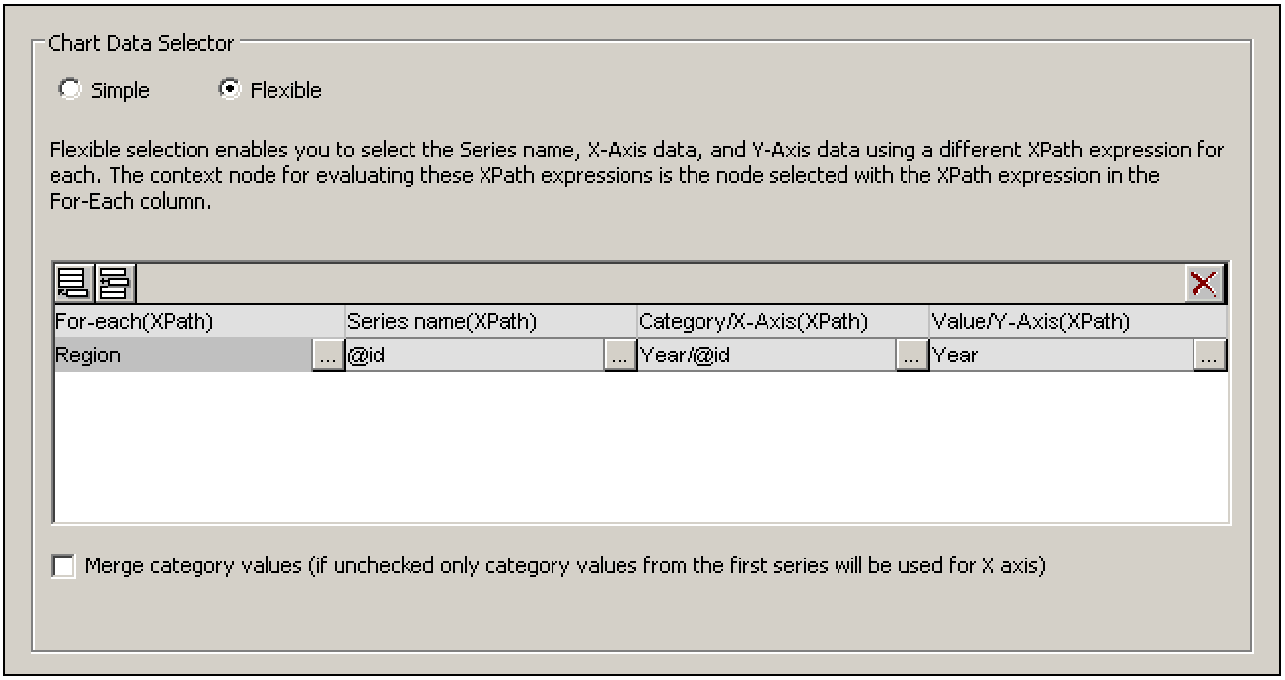


Chart Data Selector

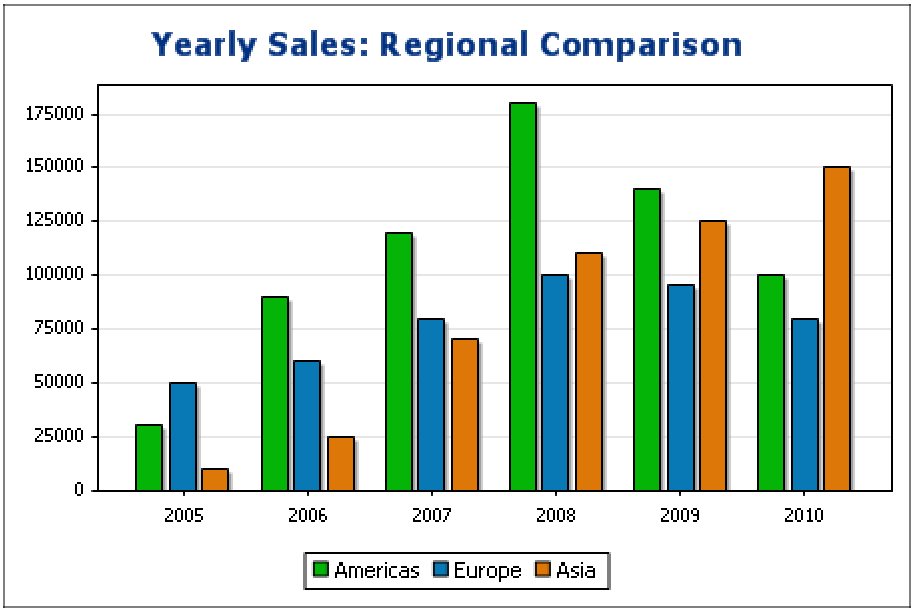
○ Simple ● Flexible

Flexible selection enables you to select the Series name, X-Axis data, and Y-Axis data using a different XPath expression for each. The context node for evaluating these XPath expressions is the node selected with the XPath expression in the For-Each column.

| For-each(XPath) | Series name(XPath) | Category/X-Axis(XPath) | Value/Y-Axis(XPath) |
|---|---|---|---|
| Region | @id | Year/@id | Year |

□ Merge category values (if unchecked only category values from the first series will be used for X axis)

For each series, the series name, X-Axis, and Y-Axis selections will correspond to the different regions because each series has a different `Region` element as its context node. The chart for this data selection will look something like this:



## Rules for chart data selection

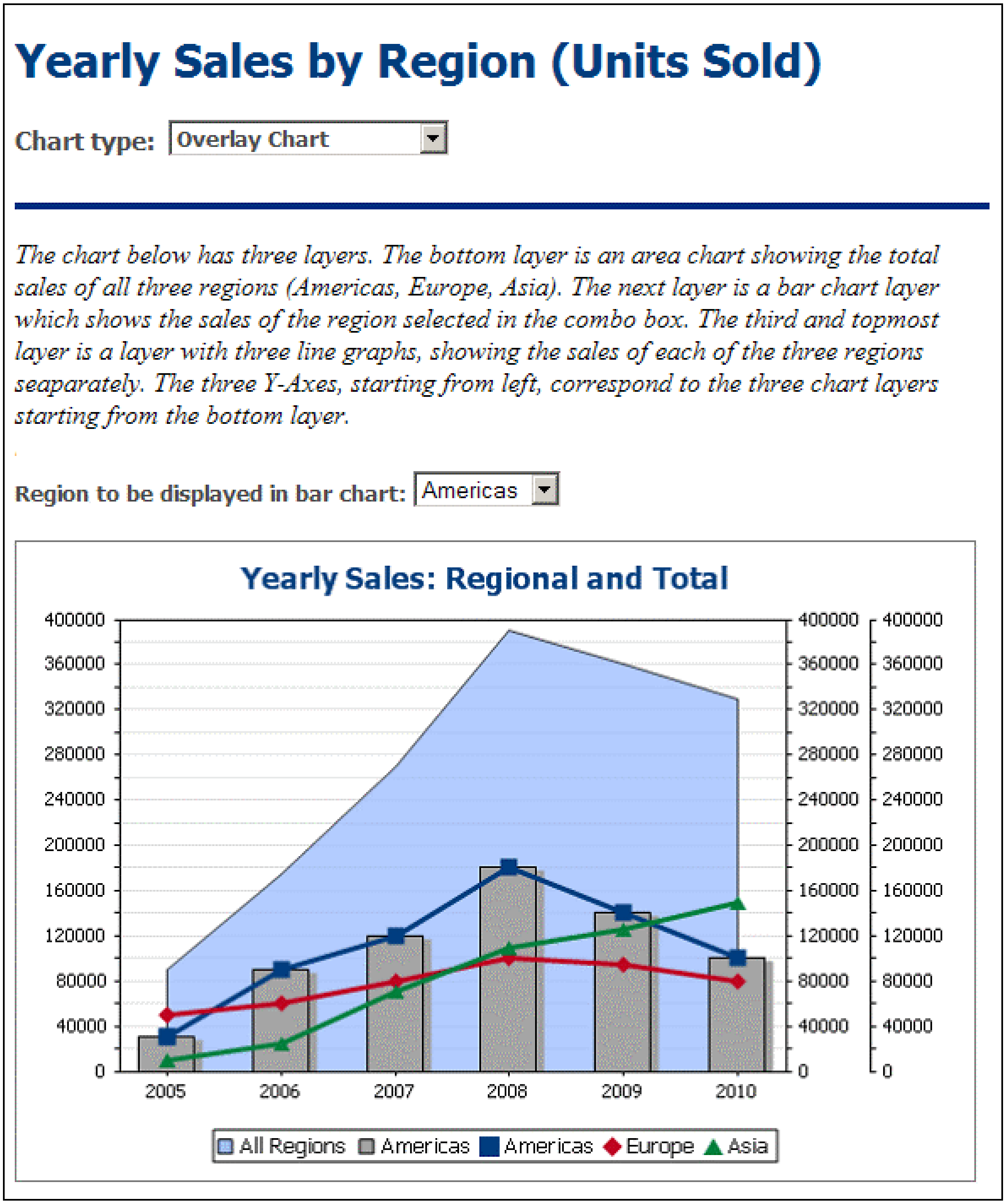
The following points should be noted when using the Chart Data Selector to select data for the various chart axes:

1. The number of bars (or pie chart slices, etc) is equal to the number of items in the larger of the X-Axis or Y-Axis sequences of a single data row selection. So if the X-Axis (which gives labels) has five items and the Y-Axis (which gives values) has six items, then six bars will be plotted with the last one being unlabeled. If the X-Axis has six items and the Y-Axis five items, then six bars will be plotted with the last one being labeled but having a value of zero.
2. The number of series is equal to the cumulative number of items in all the sequences returned by expressions in the For-Each column.
3. The name of a series is selected with the XPath expression of the Z-Axis (or Series Name Axis). If, in a data selection row, this XPath expression is left empty, then a no-name series is created. Also if the XPath expression returns a sequence with a lesser number of items than the number of series, then some series will have no name.

# 12.9.2.3 Overlay Charts

Overlay charts (*screenshot below*) enable you to combine multiple charts of different types. The overlay chart in the screenshot below is explained in the text of the screenshot. The following types of chart can be combined:

- 2D bar charts and stacked bar charts
- Line charts
- Area charts and stacked area charts
- Candlestick charts



Yearly Sales by Region (Units Sold)

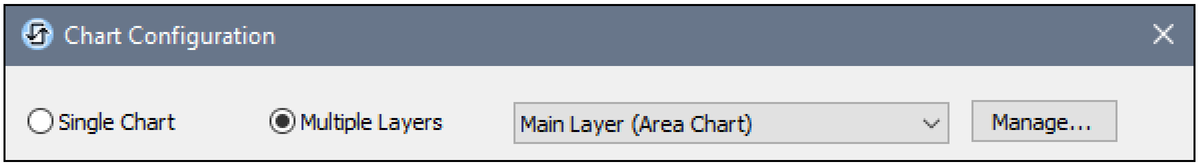
Chart type: Overlay Chart

*The chart below has three layers. The bottom layer is an area chart showing the total sales of all three regions (Americas, Europe, Asia). The next layer is a bar chart layer which shows the sales of the region selected in the combo box. The third and topmost layer is a layer with three line graphs, showing the sales of each of the three regions seaparately. The three Y-Axes, starting from left, correspond to the three chart layers starting from the bottom layer.*

Region to be displayed in bar chart: Americas

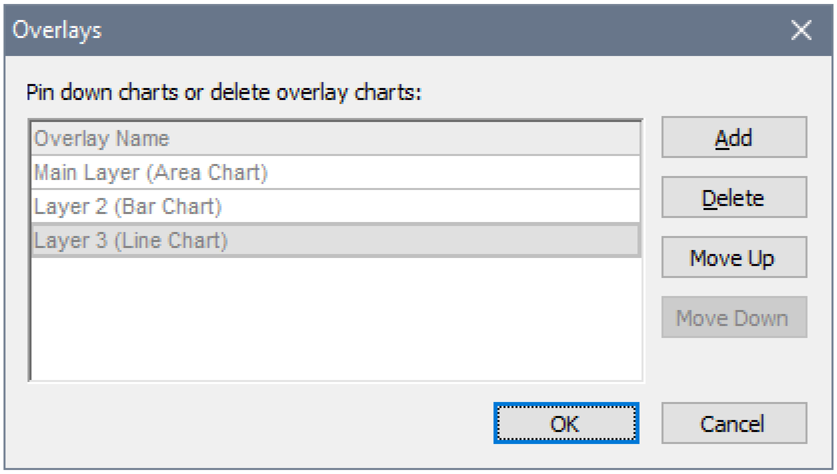
## Creating the overlay chart

An overlay chart is created by selecting the *Multiple Layers* radio button in the Chart Configuration dialog (*see screenshot below*). A main layer is automatically created. You can select the type of chart for the main layer and then make the chart settings and data selection in the usual way.



## Managing the layers of the overlay chart

You can add new layers to an overlay chart and delete layers. To do this, click the **Manage** button in the Chart Configuration dialog (*see screenshot above*). This displays the Overlays dialog (*screenshot below*). Click **Add** to add a new layer. To delete a layer, select a layer and click **Delete**.



You should note that when a layer is placed over another, it will obscure the layers beneath it. Since only area charts can be made transparent, some layer arrangements might not be optimal. For example a bar chart that is layered over a line chart would obscure parts of the lines. You should keep this in mind when planning the layering order.

## Data selection for the various charts

The data selection for each chart depends on the chart type and is done in the usual ways for each chart type. See the sections, Chart Data Selection: Simple[1156] and Chart Data Selection: Flexible[1161] for an overview of chart data selection.

The following general points relating to overlay charts are important and should be noted:

- The X-Axis for all chart layers is common and is the X-Axis specified in the main layer. If this X-Axis definition cannot be evaluated for another chart, then that chart will not be drawn. X-Axes specified in chart definitions of other layers are ignored.
- The Y-Axes series for each chart are selected separately. The Y-Axes of each layer is drawn parallel to each other, starting from the left for the bottom-most layer (the main layer) and proceeding to right with each layer. Note that a label can be specified for each Y-Axis, in its chart settings.

### Important chart settings

The following chart settings are relevant:

- The title of the chart is specified in the Chart Title setting of the main layer.
- X-Axis settings are specified in the settings for the main layer.
- Y-Axis labels can be set for each chart separately.
- Y-Axis start points (minimum) and end points (maximum), as well as the tick intervals, can be specified for each chart separately. This enables axes to be calibrated relative to each other.
- The colors of series in each chart can be selected separately.

## 12.9.3 Chart Settings and Appearance

Chart settings are organized as follows:

- Basic Chart Settings[1171], which enable you to select the type of chart and its title. Basic chart settings are defined in the Chart Settings pane of the Chart Configuration dialog (*see screenshot below*).
- Advanced Chart Settings[1178],which enable you to change the appearance of a chart (its title, legend, colors, fonts, etc). Advanced settings are defined in the Change Appearance dialog[1178]. To access this dialog, click **All Settings** in the Chart Configuration dialog (*see screenshot below*).
- Dynamic XPath Settings[1198], which can be accessed by clicking **Dynamic XPath Settings** in the Chart Settings pane (*see screenshot below*).

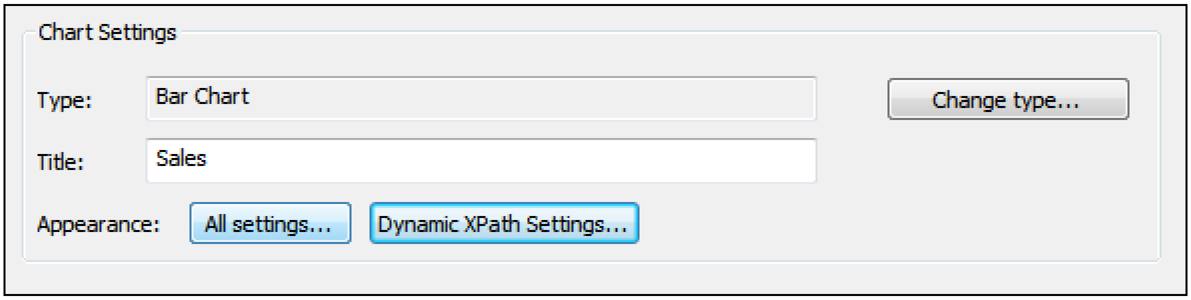


Chart Settings

Type: Bar Chart

Title: Sales

### 12.9.3.1 Basic Chart Settings

*This section:*

- Setting the chart type[1172]
- List of chart types[1172]
- Other basic settings[1177]

## Setting the chart type

The most basic chart setting is the chart type. To select the chart type, click **Change Type** in the Chart Settings pane 1171 of the Chart Configuration dialog.

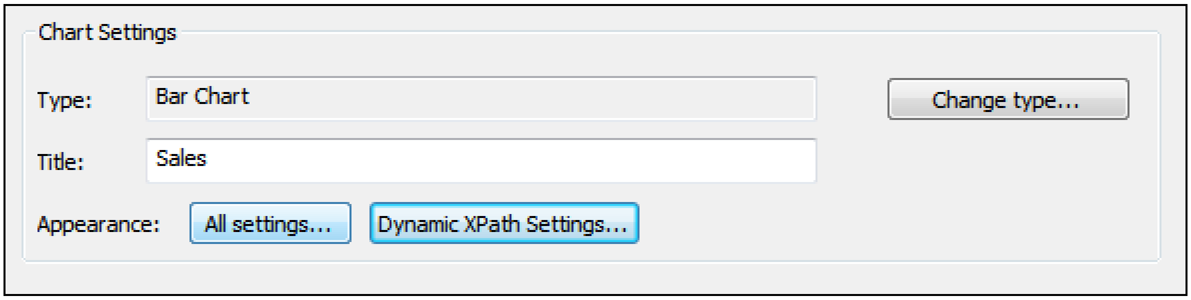
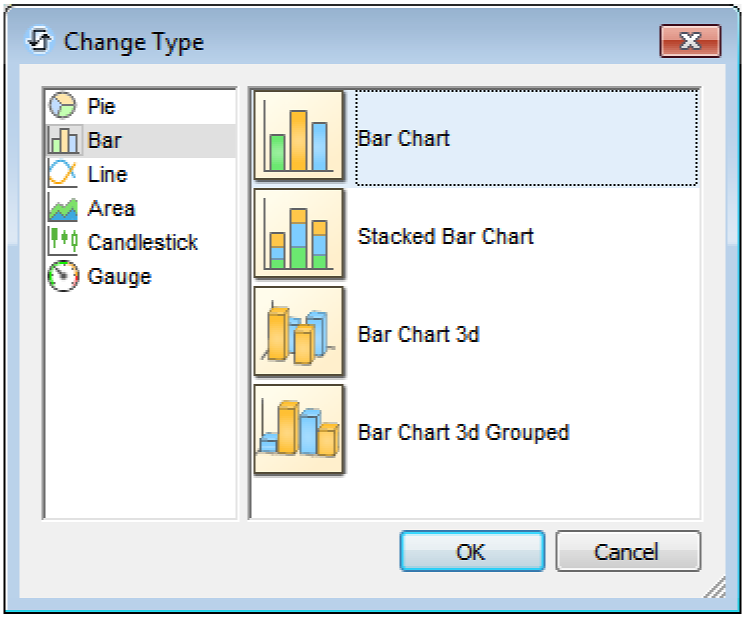


Chart Settings

Type: Bar Chart

Title: Sales

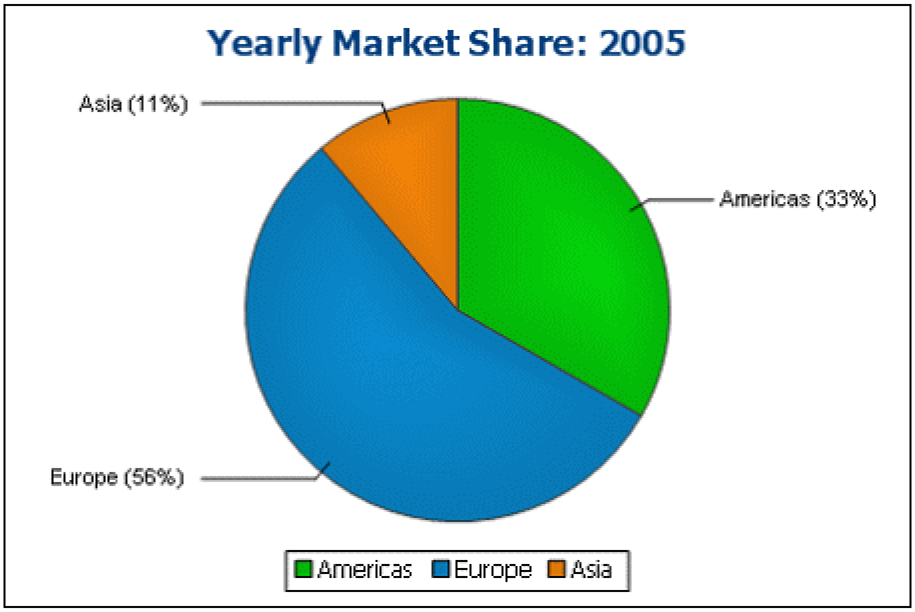


## Chart types

The various types of charts that are available are listed below. In the Change Type dialog 1171 (*screenshot above*), select the chart type you want and click **OK**.

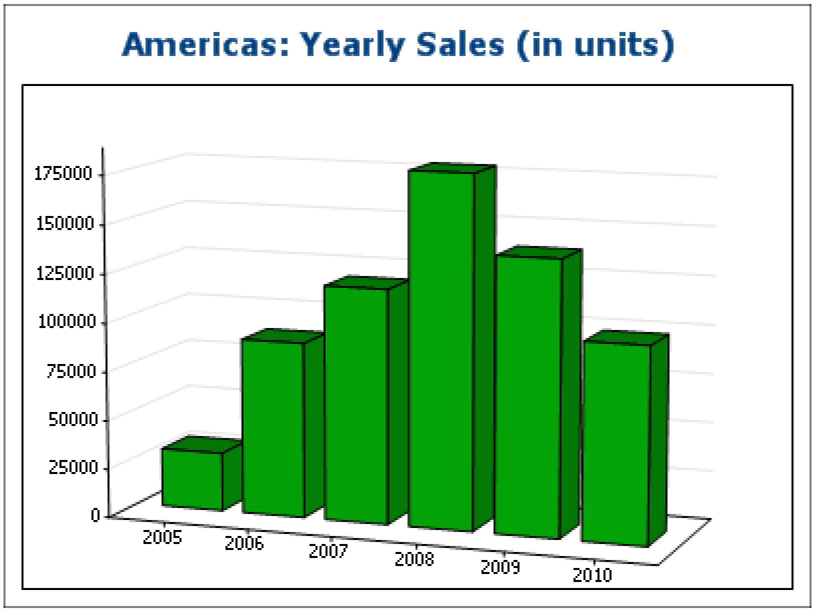
▼ *Pie charts*

In pie charts, one column/axis provides the values, another column/axis provides labels for these values. The labeling column/axis can take non-numeric values.

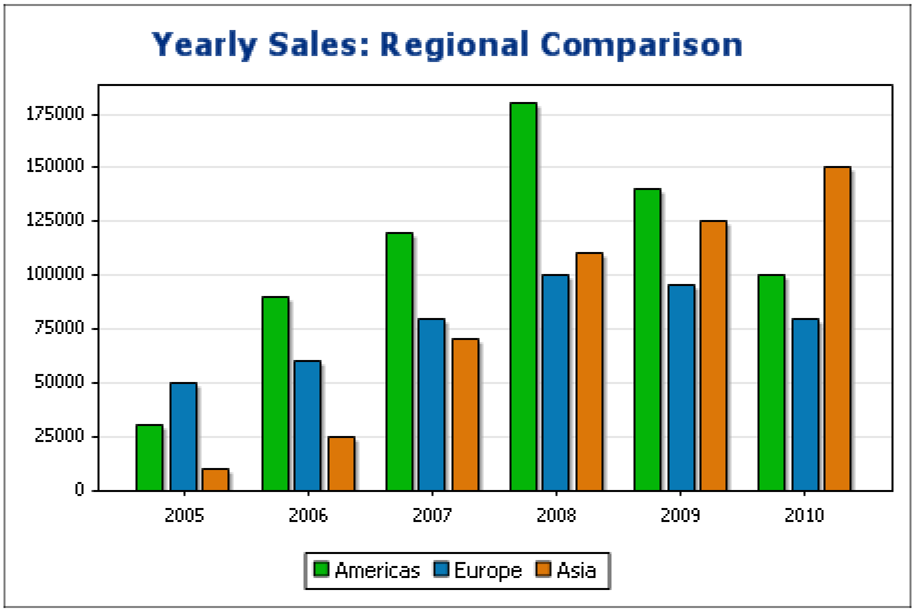


▼ *Bar charts*

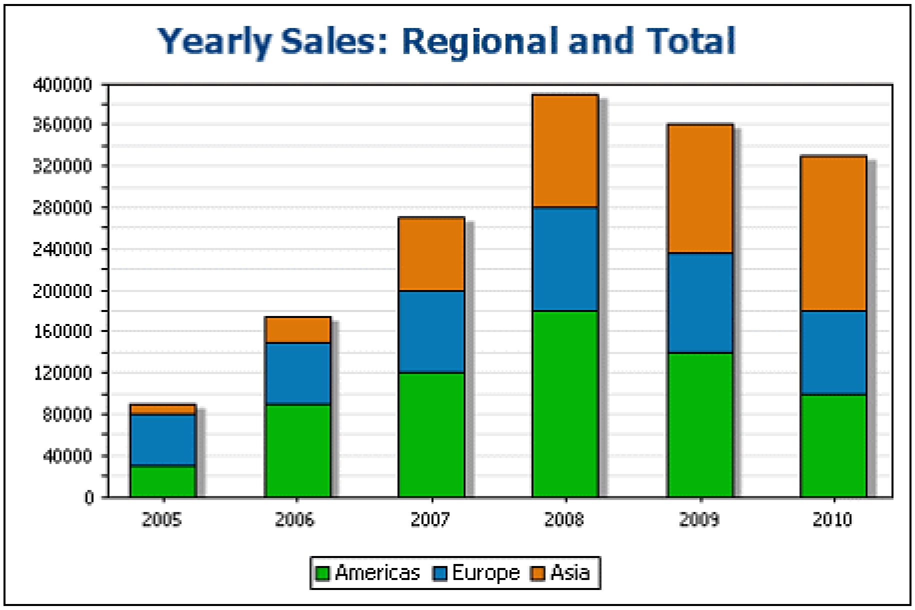
Bar charts can have two sets of values used along two axes (*below*).



They can also use three sets of values, as in the example below: (i) continent, (ii) year, (iii) sales volume. Bar charts can be displayed in 2D (*below*) or 3D (*above*).

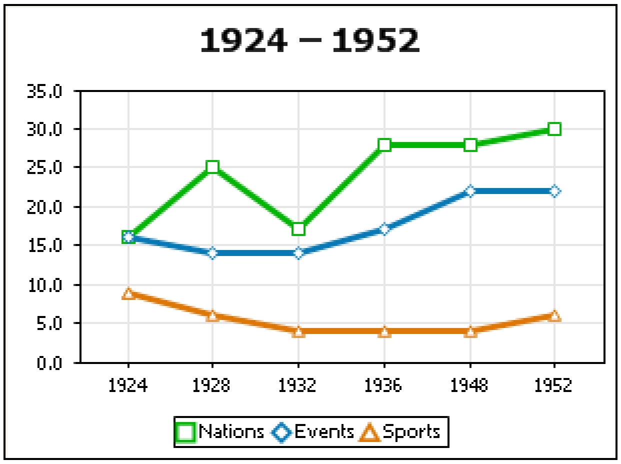


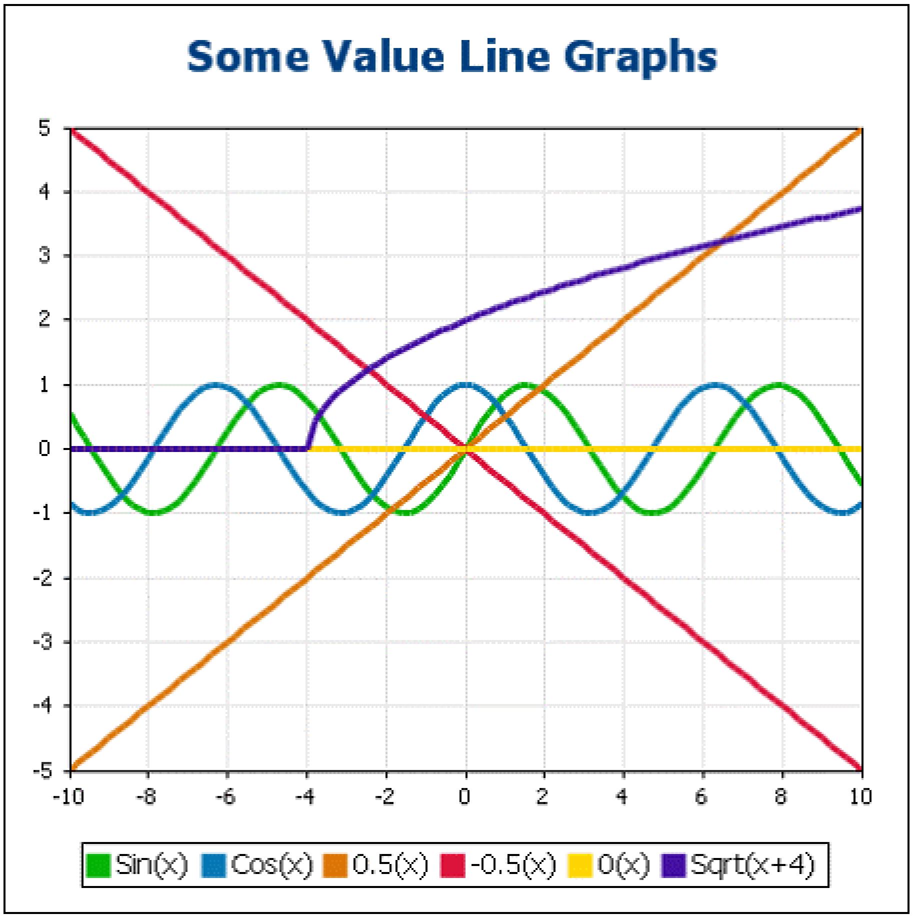
A three-axis bar chart can also be stacked if you need to show totals. Compare the stacked chart below with the chart above. The stacked chart shows the total of sales on all continents.



▼ *Line charts*

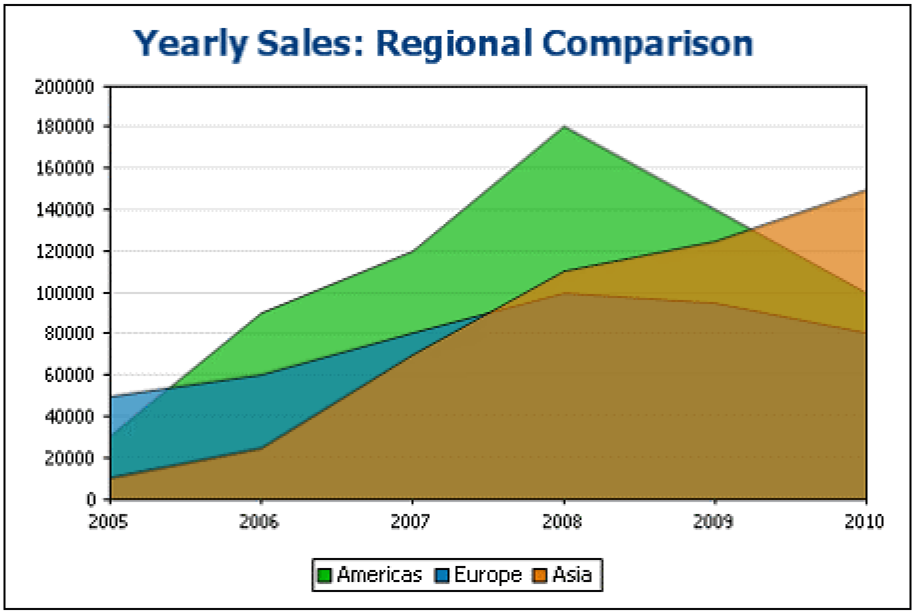
The difference between a line chart (*below left*) and a value line chart (*below right*) is that value line charts only take numerical values for the X-axis. If you need to display line charts with text values on the X-axis, use line charts.





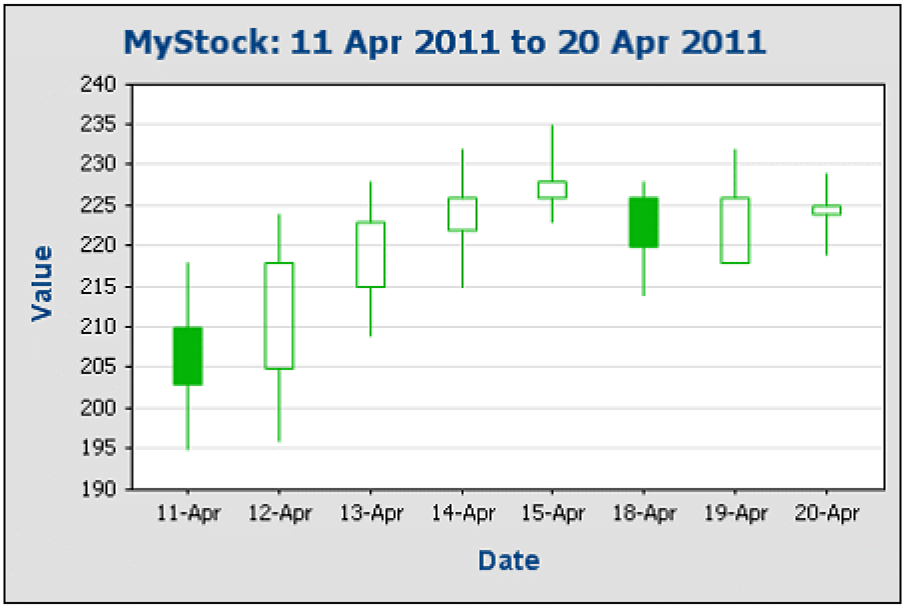
▼ *Area charts*

Area charts are a variation of line charts, in which the areas below the lines are also colored. Note that area charts can also be stacked (*see bar graphs above*).



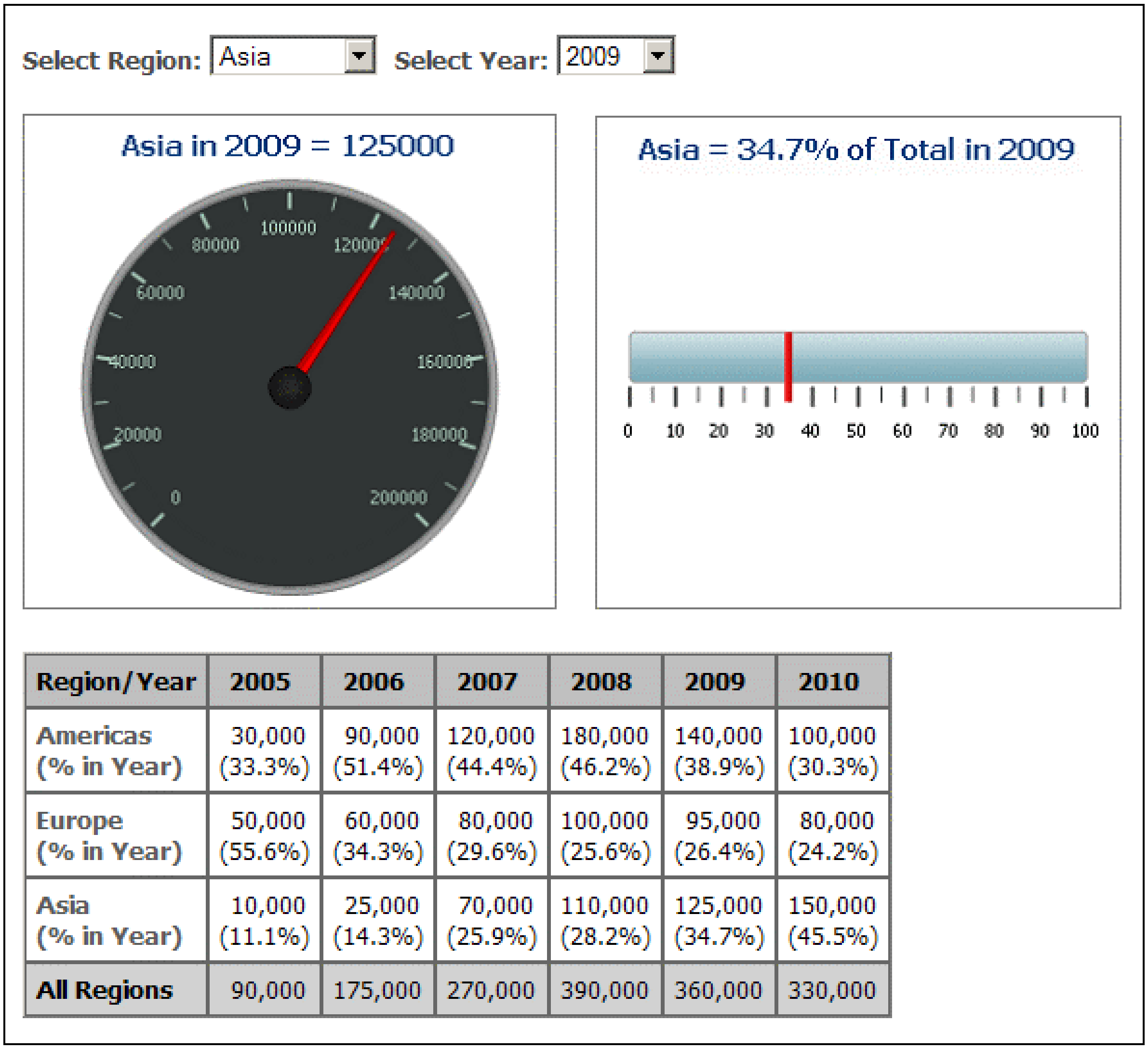
▼ *Candlestick charts*

A candlestick chart can be used to depict price movements of securities, commodities, currencies, etc over a period of time. The chart indicates not only how prices developed over time, but also the daily close, high, low, and (optionally) open. The Y-axis takes three or four series (close, high, low, and (optionally) open). The screenshot below shows a four-series candlestick chart.



▼ *Gauge charts*

Gauge charts are used to illustrate a single value and show its relation to a minimum and a maximum value.



Select Region: Asia

Select Year: 2009

Asia in 2009 = 125000

Asia = 34.7% of Total in 2009

| Region/Year | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 |
|---|---|---|---|---|---|---|
| Americas (% in Year) | 30,000 (33.3%) | 90,000 (51.4%) | 120,000 (44.4%) | 180,000 (46.2%) | 140,000 (38.9%) | 100,000 (30.3%) |
| Europe (% in Year) | 50,000 (55.6%) | 60,000 (34.3%) | 80,000 (29.6%) | 100,000 (25.6%) | 95,000 (26.4%) | 80,000 (24.2%) |
| Asia (% in Year) | 10,000 (11.1%) | 25,000 (14.3%) | 70,000 (25.9%) | 110,000 (28.2%) | 125,000 (34.7%) | 150,000 (45.5%) |
| **All Regions** | 90,000 | 175,000 | 270,000 | 390,000 | 360,000 | 330,000 |

## Other basic settings

In the Chart Settings pane, you can also set the title of the chart (*see screenshot below*).

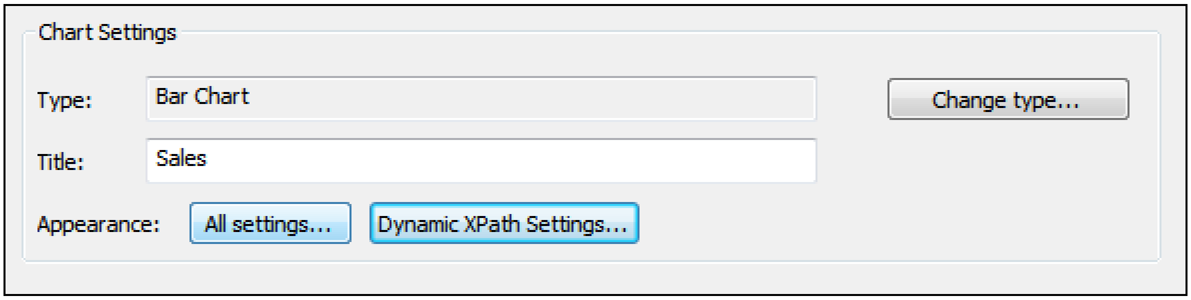


Chart Settings

Type: Bar Chart

Title: Sales

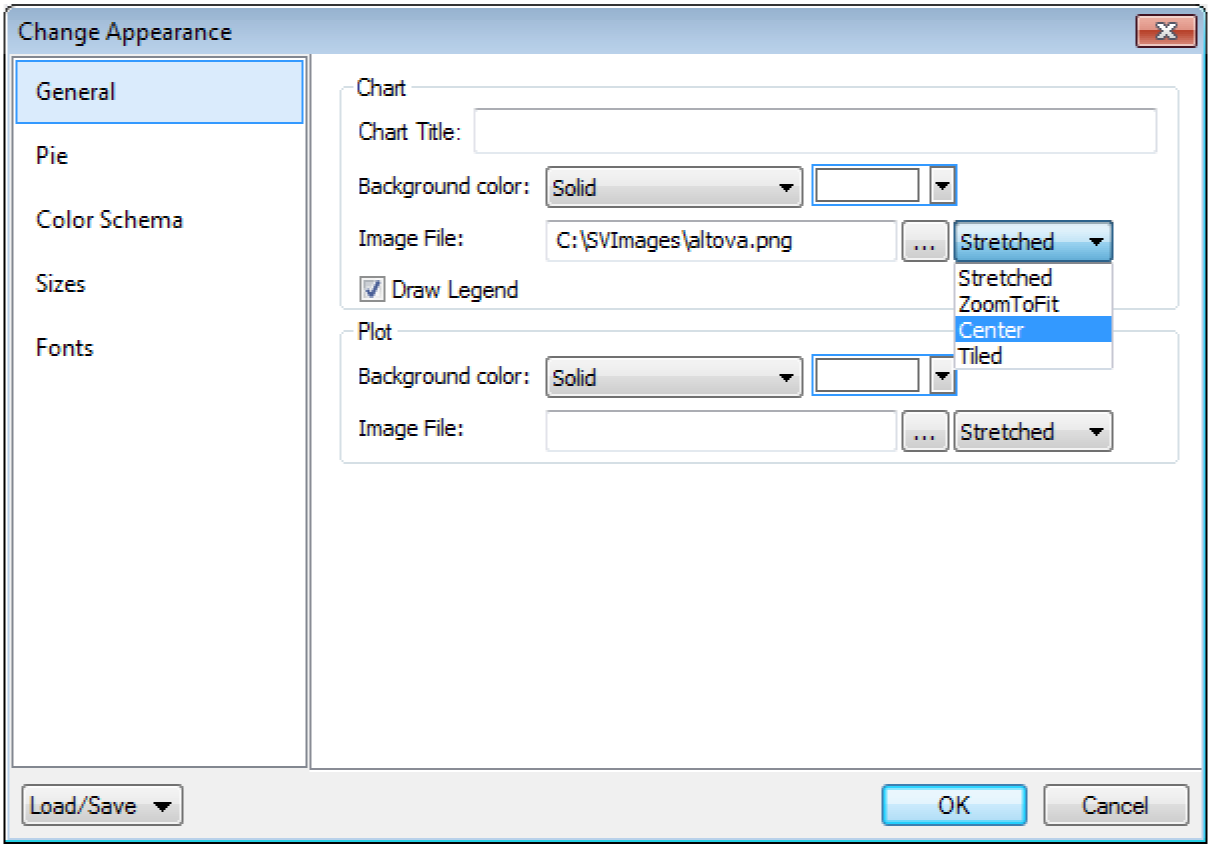
## 12.9.3.2 Advanced Chart Settings

*This section*:

- Accessing the advanced settings 1178
- Overview of advanced settings 1179
- Loading, saving, resetting chart settings 1181

### Accessing the advanced settings

To access a chart's advanced settings do the following: Click **All Settings** 1171 in the Chart Configuration dialog. This displays the Change Appearance dialog for that particular chart type (*the screenshot below shows the Change Appearance dialog of a pie chart*).



Chart

Chart Title:

Background color: Solid

Image File: C:\SVImages\altova.png ... Stretched

☑ Draw Legend

Plot

Background color: Solid

Image File: ... Stretched

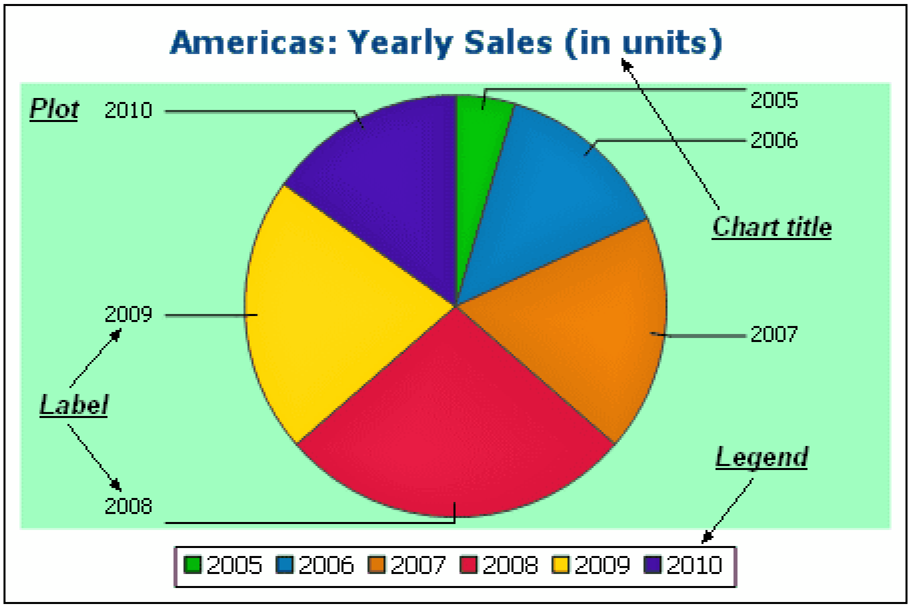
## Overview of advanced settings

The advanced settings are organized into tabs that are common to all chart types and those that are specific to a single chart type.

*Common chart settings*

▼ *General*

The chart title (*see screenshot below*) is the same as the basic setting (*see above*) and can be edited as an advanced setting also. Other settings in this dialog are the background color of the chart and the plot. In the screenshot below, the plot has been given a pale green background color. An image file can also be set as the background image of the chart and/or the plot. This image can be stretched to cover the entire area of the chart or plot; zoomed to fit so that the zoom matches one of the two dimensions (of chart/plot); centered; or tiled. The legend is the key to the color codes in the chart, and it can be turned on or off.



▼ *Color scheme*

Four predefined color schemes are available plus a user-defined color scheme. You can modify any of the color schemes by adding colors to and/or deleting colors from a scheme. The color scheme selected in this tab will be used in the chart.

▼ *Sizes*

Sizes of various aspects of the chart can be set, either as pixels or as a percentage ratio.

▼ *Font*

The font properties of the chart title and of legends and labels can be specified in this tab. Sizes can be set as a percentage of the chart size or absolutely as points.

▼ *Load/Save button*

Settings can be saved to an XML file and can be loaded from an XML file having the correct structure. To see the structure, save the settings of a chart and then open the XML file. Clicking this button also gives you the option of resetting chart settings to the default.

*Type-specific chart settings*

▼ *Pie charts*

Settings for: (i) the angle from which the first slice should be drawn; (ii) the direction in which slices should be drawn; (iii) the outline color; (iv) whether the colors receive highlights (in 3D pie charts: whether dropshadows and transparency are used); (v) whether labels should be drawn; and (vi) whether values and percentages should be added to labels and how many decimal places should be added to the percentages.

▼ *Bar charts*

Settings for: *(General)* Drawing the X and Y axes exchanged generates a horizontal bar chart; *(Bar)* Bar outlines and dropshadows (dropshadows in 2D bar charts only); *(X-Axis)* Label and color of the x-axis, and vertical gridlines; *(Y-Axis)* Label and color of the y-axis, horizontal gridlines, the range of values to be displayed, and the tick marks on the y-axis; *(Z-Axis, 3D only)* Label and color of the z-axis; *(3D)* the vertical tilt, horizontal rotation, and the width of the view.

▼ *Line graphs*

Settings for: *(General)* Drawing the X and Y axes exchanged; *(Line)* including the plot points or not; *(X-Axis)* Label and color of the x-axis, and vertical gridlines; *(Y-Axis)* Label and color of the y-axis, horizontal gridlines, the range of values to be displayed, and the tick marks on the y-axis.

▼ *Gauge charts*

Settings for: (i) the angle at which the gauge starts and the angular sweep of the scale; (ii) the range of the values displayed; (iii) the interval and color of major and minor ticks; (iv) colors of the dial, the needle, and the border.

▼ *Area charts*

The transparency of areas can be set as a value from 0 (no transparency) to 255 (maximum transparency). In the case of non-stacked area charts transparency makes parts of areas that lie under other areas visible to the viewer. Outlines for the areas can also be specified.

▼ *Candlestick charts*

The fill color can be specified for the two situations: (i) when the closing value is greater than the opening value, and (ii) when the opening value is greater than the closing value. In the latter case, the Series color is also available as an option. The Series color is specified in the Color Schema tab of the Change Appearance dialog.

## Loading, saving, resetting chart settings

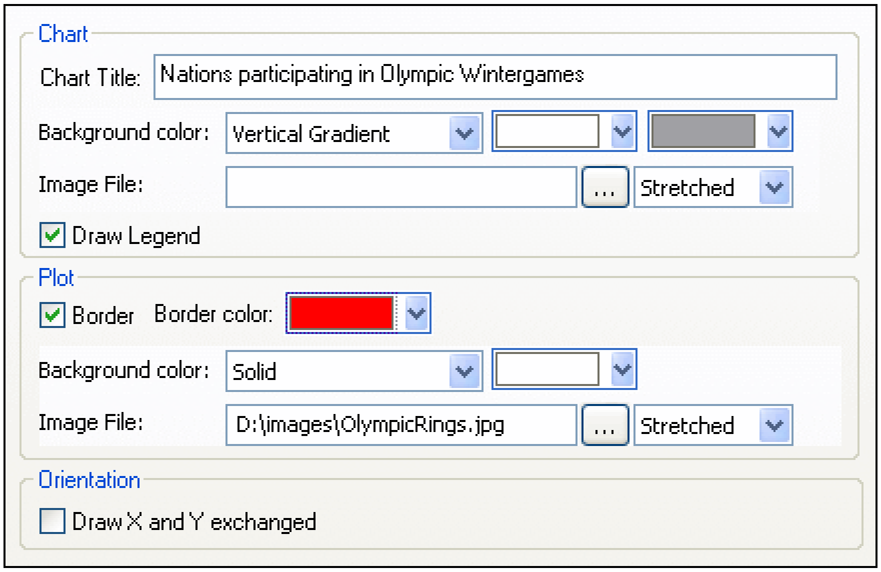
Chart settings that are different from the default settings can be saved in an XML file. These settings can subsequently loaded as the settings of a chart, which can help you save time and effort. The **Load/Save** button (see first screenshot in this section[1178]) provides the following options when clicked:

- **Set to default**: Rejects changes made to the settings, and restores the default settings to **all** settings sections.

- **Load from file**: Enables settings to be imported that have been previously saved in an XML file (*see next command*). The command displays the Open dialog, in which you enter the location of the required file.
- **Save to file**: Opens the Save As dialog box. You can specify an XML file in which to save the settings. This file lists those settings that are different from the default settings.

### 12.9.3.2.1 General

The General section of the Change Appearance dialog box lets you define the title of the chart, add or remove a legend, and define background pictures and colors and—for bar, line, area, and candlestick charts—orientation of the chart.



Chart

Chart Title: Nations participating in Olympic Wintergames

Background color: Vertical Gradient

Image File: ... Stretched

☑ Draw Legend

Plot

☑ Border Border color:

Background color: Solid

Image File: D:\images\OlympicRings.jpg ... Stretched

Orientation

□ Draw X and Y exchanged

## Chart

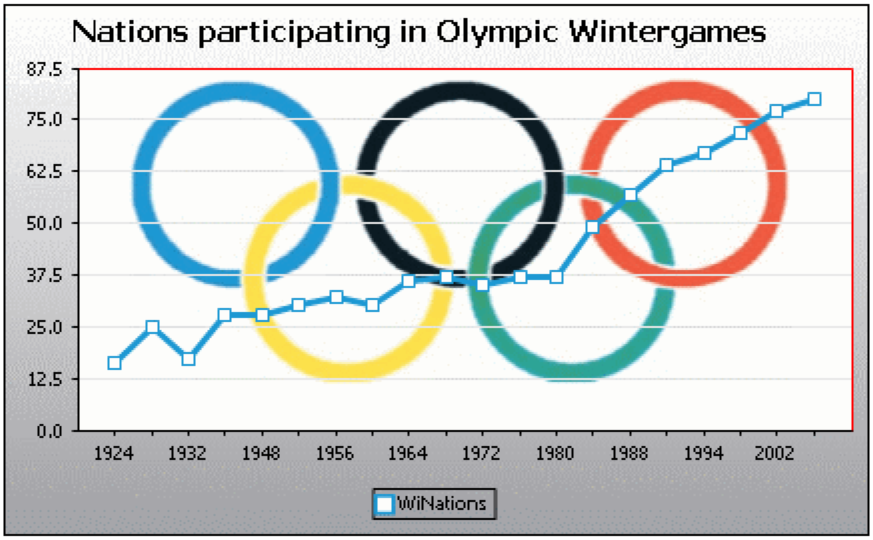
Enter a descriptive title for your chart into the `Chart Title` field and select a background color for the entire chart from the drop-down list. You can choose a solid background, vertical gradient, or horizontal gradient and define start and end colors for the gradient, if applicable. In addition, or instead of a colored background, you can also define a background image and choose one of the available display options from the drop-down list:

- Stretched: the image will be stretched to the height and width of the chart
- Zoom to Fit: the image will be fit into the frame of the chart and the aspect ratio of the image will be maintained
- Center: the image will be displayed in its original size in the center of the chart
- Tiled: if the image is smaller than the chart, duplicates of the image will be displayed to fill the background area

The `Draw Legend` check box is activated by default, clear the check box if you do not want to display a legend in your chart.

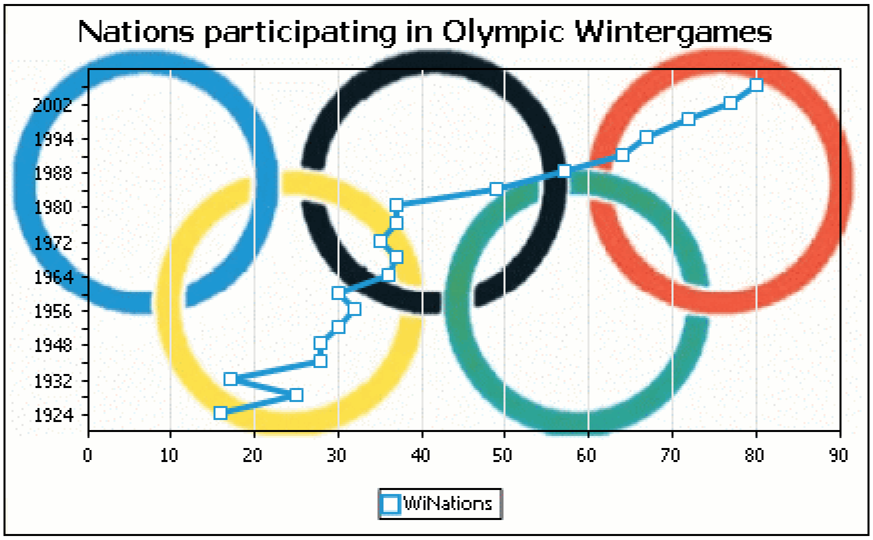
## Plot

The Plot is the area where the actual data of the chart is displayed. You can draw a border around the plot and specify a different background color and/or image for the plot area. In the screenshot below, the background color of the chart has been changed to gray (vertical gradient) whereas the plot is still white, a red border has been drawn around it, and a background image has been added.



## Orientation

If you have a small series of large values it may be convenient to swap the X and Y axis for a better illustration. Note that in the screenshot below also the background color of the plot has been set to "Transparent" and the background image has been applied to the chart.



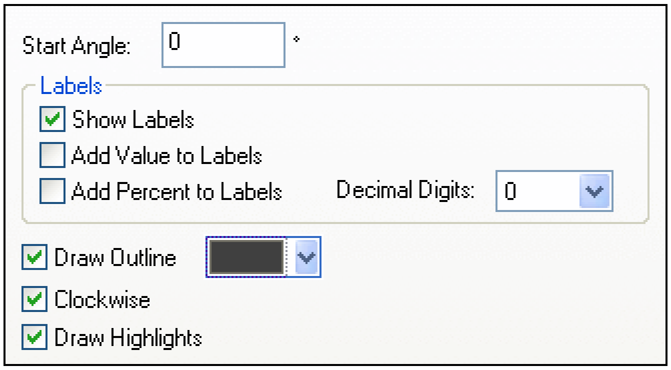
Note that this option is not available for pie and gauge charts.

## 12.9.3.2.2 Type-Related Features

For each of the chart types, and even for the various sub-types, the Change Appearance dialog box provides a section where you can define the type-related features of the chart.

### Pie chart

Most settings are the same for the 2d and 3d versions. In 2d pie charts, you can additionally draw highlights.



Start Angle: 0 °

Labels

☑ Show Labels

□ Add Value to Labels

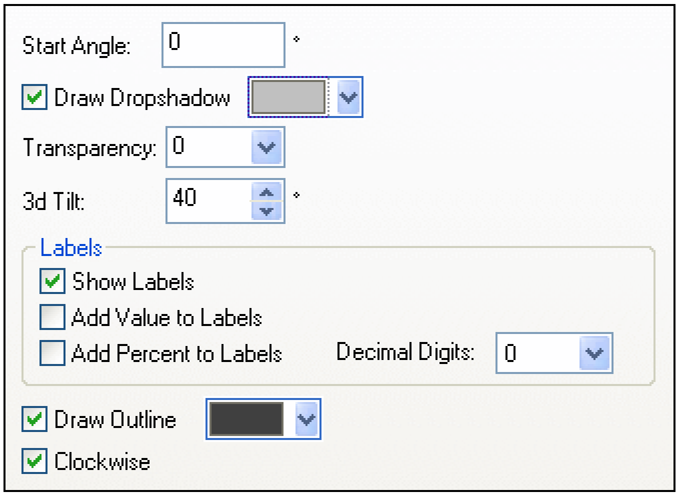
□ Add Percent to Labels Decimal Digits: 0

☑ Draw Outline

☑ Clockwise

☑ Draw Highlights

In 3d pie charts, you can display drop shadows, add transparency and define the 3d tilt.



Start Angle: 0 °

☑ Draw Dropshadow

Transparency: 0

3d Tilt: 40 °

Labels

☑ Show Labels

□ Add Value to Labels

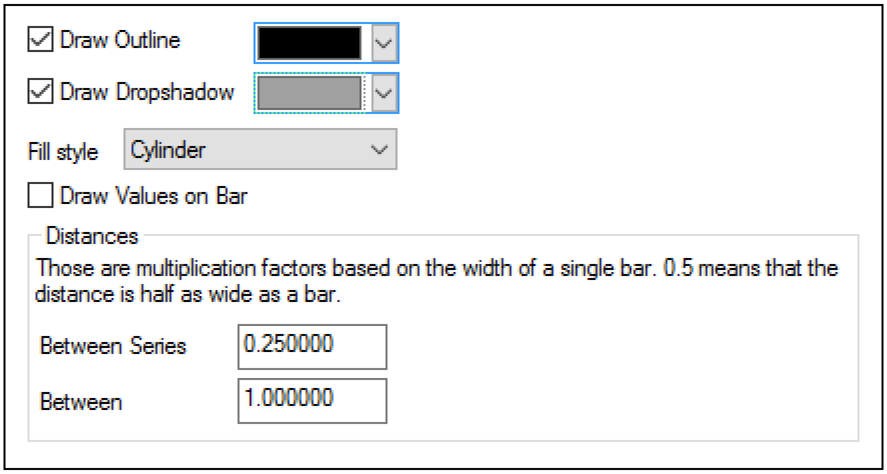
□ Add Percent to Labels Decimal Digits: 0

☑ Draw Outline

☑ Clockwise

The `Start Angle` value defines where the first row of the selected column will be displayed in the chart. An angle of 0 degrees corresponds to 12 o'clock on a watch.
You can show labels in addition to, or instead of, the legend, add values and/or percentage to the labels, and define for the percentage values the number of decimal digits to be displayed.
The color that you can select next to the `Draw Outline` check box is used for the optional border drawn around the chart and the individual pie segments. The `Clockwise` check box allows you to specify whether the rows should be listed clockwise or counter-clockwise.
In 3d pie charts, you can draw a drop shadow and define its color, add transparency to the chart, and define the 3d tilt. In 2d charts, the `Draw Highlights` option adds additional structure to the chart.

## Bar chart



☑ Draw Outline

☑ Draw Dropshadow

Fill style: Cylinder

□ Draw Values on Bar

Distances

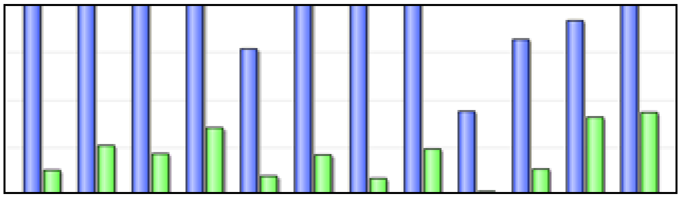
Those are multiplication factors based on the width of a single bar. 0.5 means that the distance is half as wide as a bar.

Between Series: 0.250000

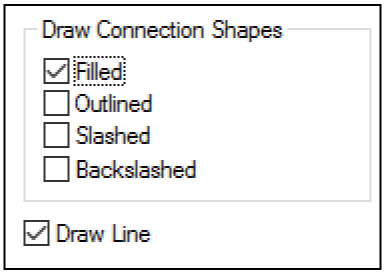
Between: 1.000000

For bar charts, you can make the following setting:

- Add an outline to the bars and define its color.
- In 2d bar charts, you can also draw a drop shadow and define its color (this option is not available for 3d bar charts).
- By default, the shape of the bars resembles a cylinder, however you can also choose "Vertical Gradient" or "Solid" from the `Fill style` drop-down list (this option is available for 2d bar charts only).
- The values of a bar (corresponding to the height of the bar on the Y-axis) can be drawn on the bar. The font of the values can be specified in the *Fonts* settings (this option is available only for 2d bar charts, not stacked).
- The distance between the series of a bar-group and between bar-groups can be specified as a decimal fraction of the width of a single bar. For example, in the screenshot below, which shows bar-groups that each consist of a blue series and a green series, the distance between the series has been set to a 25% (**`=0.25`**) of the width of a bar; the distance between bar-groups has been set to 100% (**`=1.0`**) of the width of a bar. This option is available only for 2d bar charts.



## Line chart

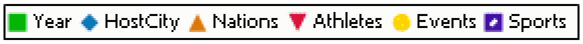


Draw Connection Shapes
☑ Filled
□ Outlined
□ Slashed
□ Backslashed

☑ Draw Line

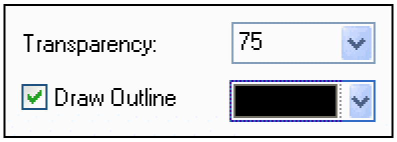
To draw connection shapes that mark the values in line charts, you need to activate at least one check box in the Draw Connection Shapes group box. You can use five different shapes to mark a series: square, rhomb, triangle, inverted triangle, and circle. If there are more than five series in your chart you can combine the connection shapes by selecting more than one option in the Draw Connection Shapes group box. In the screenshot below, both `Filled` and `Slashed` have been selected and the `Slashed` type is used for the sixth series and beyond.

The *Draw Line* option enables the graph to be drawn with (i) only connection shapes, or (ii) with connection shapes joined by a line.



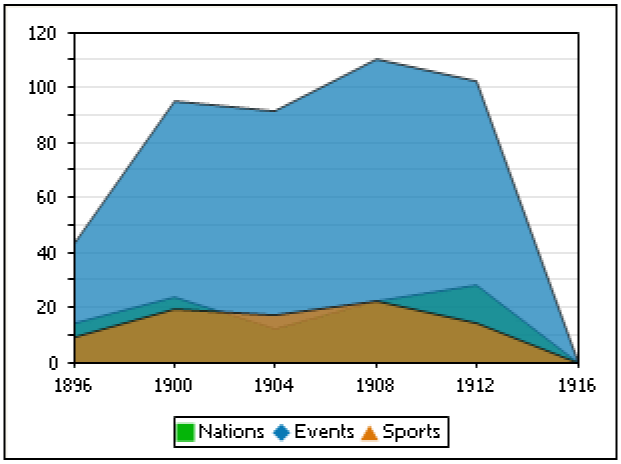
Connection shapes are available for both line charts and value line charts.

## Area chart

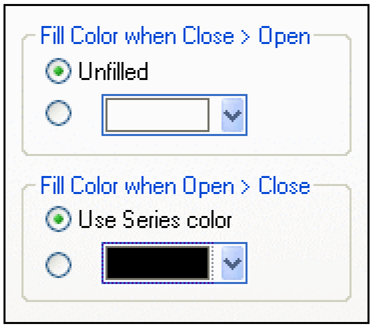


Transparency: 75
☑ Draw Outline ■

Among the properties that you can change for area charts is transparency; this way you can prevent that one series is hidden by another series in the chart. In addition, you can add an outline to the individual data areas and define its color (*see screenshot below*).

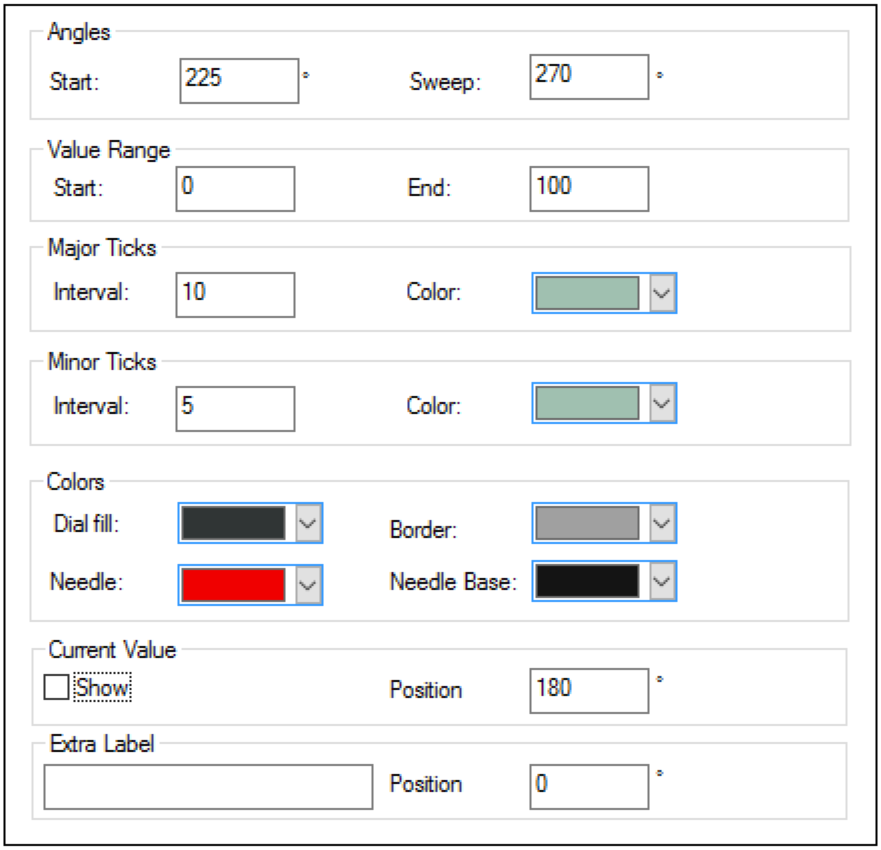


## Candlestick chart



If both opening and closing value are defined as series, you can choose the colors and whether or not the candle should be filled if the closing value is greater than the opening value.

## Gauge chart



Angles
Start: 225 ° Sweep: 270 °

Value Range
Start: 0 End: 100

Major Ticks
Interval: 10 Color: 🟩

Minor Ticks
Interval: 5 Color: 🟩

Colors
Dial fill: ⬛ Border: ⬜
Needle: 🟥 Needle Base: ⬛

Current Value
☐ Show Position 180 °

Extra Label
Position 0 °

The `Start` value in the Angles group box defines the position of the 0 mark and the `Sweep` value is the angle that is used for display. In the Value Range group box you can define the minimum and maximum values to be displayed. Tick marks are displayed with (major ticks) or without (minor ticks) the corresponding value; you can define separate colors for them. In the Colors group box you can define colors for the dial fill, needle, needle base (hides the first part of the needle in the center of the chart), and the border that surrounds the chart. The current value and an extra label can be shown at any angle you like.

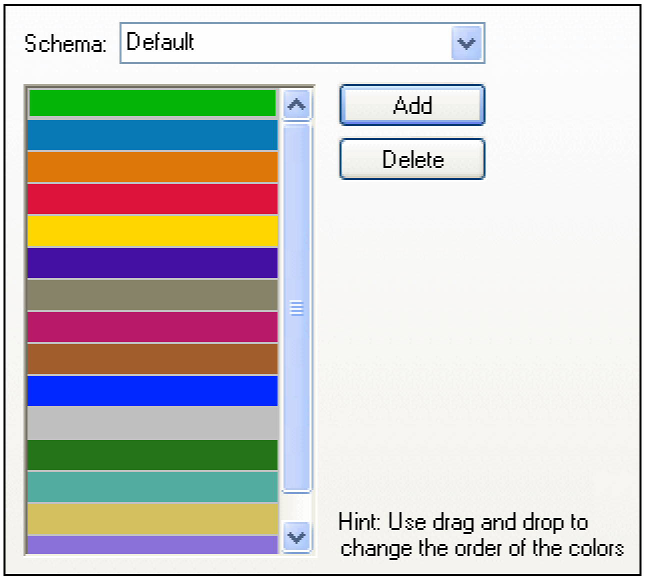
### 12.9.3.2.3 Colors

Depending on the chart type you have selected, MobileTogether Designer provides two different sections for the definition of colors to be used in charts:

- Color Schema for pie, bar, line, area, and candlestick charts
- Color Range for gauge charts

## Color Schema

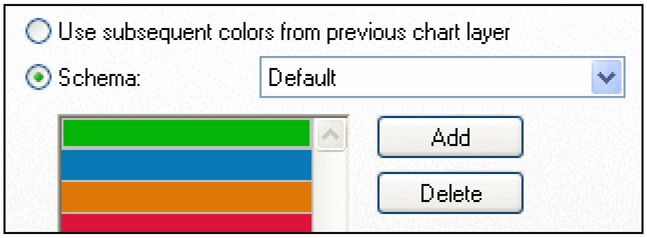
The Color Schema section of the Change Appearance dialog box provides four predefined color schemas (i.e., default, grayscale, colorful, and pastel) that can be customized; in addition you can also define your own color schema from scratch.



Hint: Use drag and drop to change the order of the colors

The top color will be used for the first series, then the second color and so on. You can change the order of the colors by selecting a color and dragging it to its new position with the mouse. To add a new or delete an unwanted color, click the corresponding button. In candlestick charts, only the first color will be used.

If you have appended one or several layers of overlay charts to a Charts window, the Color Schema section of the Change Appearance dialog box contains the additional radio button `Use subsequent colors from previous chart layer` which is activated by default.

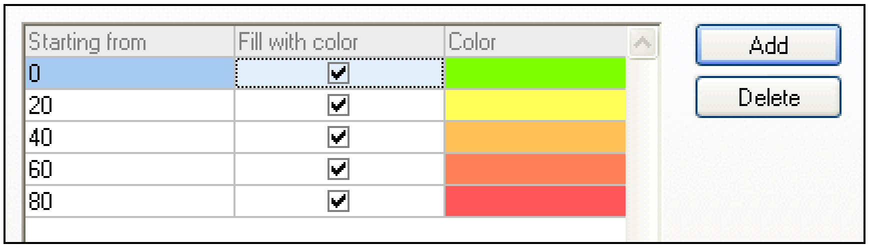


When the radio button is activated, the color schema from the previous layer will be used and you cannot choose a separate color schema for the overlay. The series of the active layer will be displayed using subsequent colors from the color schema of the previous layer. This way, all series of the Charts window have different colors and can therefore be distinguished more easily.

You can break this link on any additional layer that you add and choose a different color schema that then can also be re-used in subsequent layers.

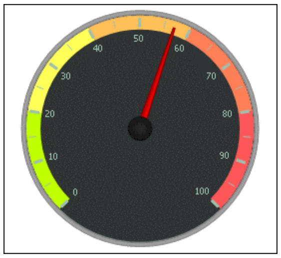
## Color Range

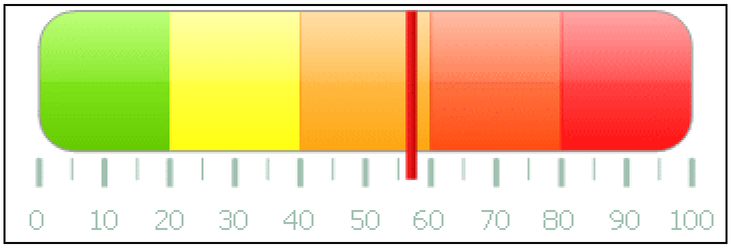
In gauge charts, you can customize the appearance of the gauge by applying colors to certain value ranges.



| Starting from | Fill with color | Color |
|---|---|---|
| 0 | ☑ | 🟩 |
| 20 | ☑ | 🟨 |
| 40 | ☑ | 🟧 |
| 60 | ☑ | 🟧 |
| 80 | ☑ | 🟥 |

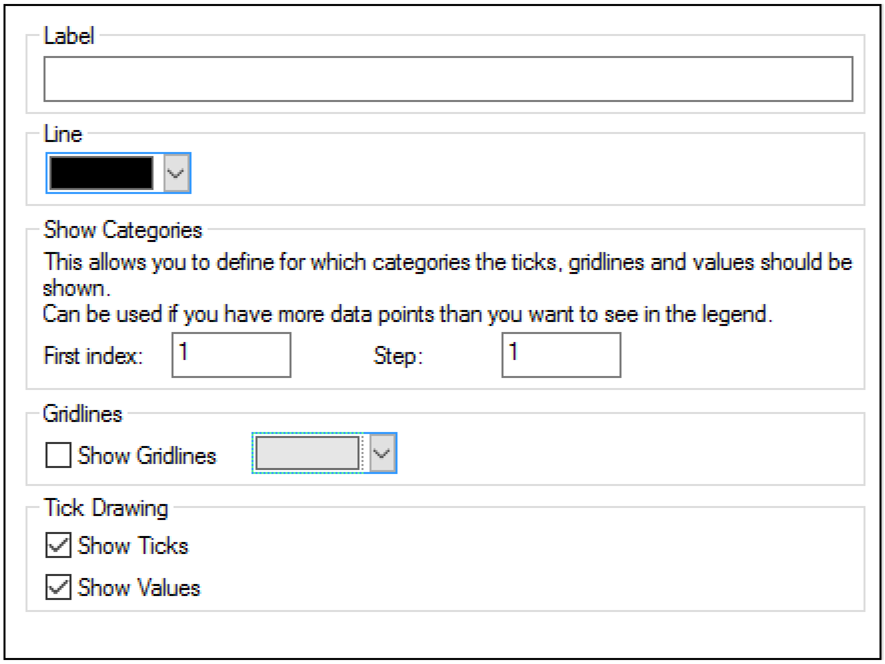
The definition shown in the screenshot above will appear in the gauge charts as follows:





### 12.9.3.2.4 X-Axis

In the X-axis section of the Change Appearance dialog box, you can enter a label for the axis, and define colors for the axis and the grid lines (if displayed). You can also define whether or not you want to display tick marks and axis values. This section is the same for all bar, line, area, and candlestick charts. The *Show Categories* options enables you to specify that only a subset of all categories (X-Axis values) are displayed, that is, only the ticks, grid lines, and values of the selected categories will be displayed. Create the subset of displayed categories by entering (i) the index of the first value to display, and (ii) the number of indices to step. For example, if there are 101 categories, from `1900, 1901, 1902 ... 1999, 2000`, then you can show every tenth year from `1900` to `2000` by setting *First index* to `1` and *Step* to `10`.



Label

Line

Show Categories

This allows you to define for which categories the ticks, gridlines and values should be shown.
Can be used if you have more data points than you want to see in the legend.

First index: 1 Step: 1

Gridlines

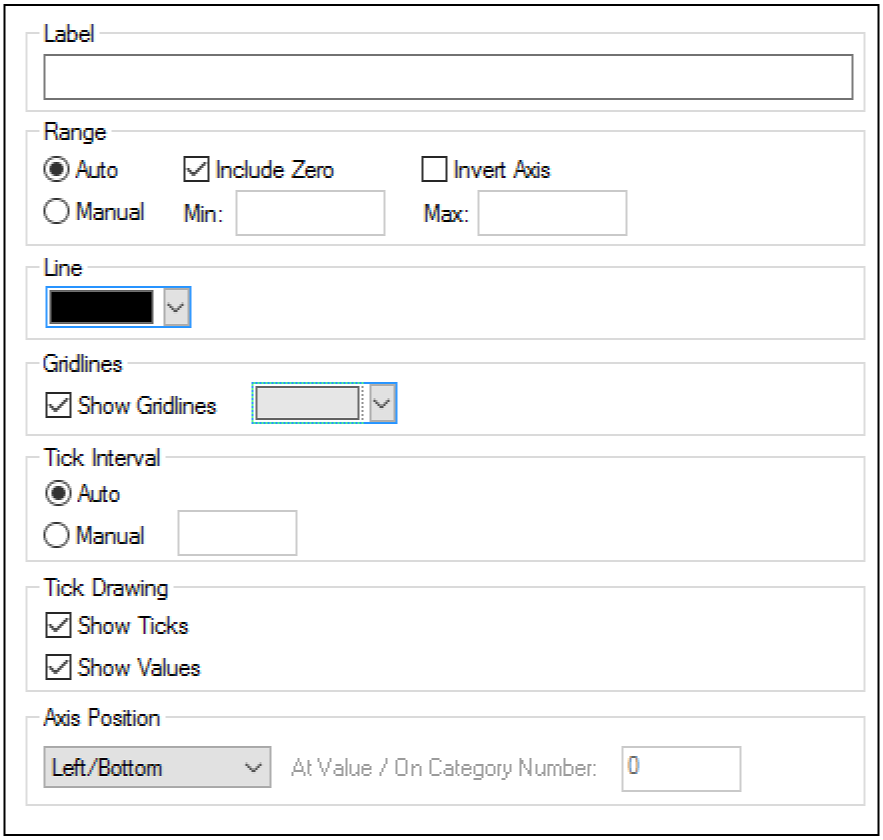
□ Show Gridlines

Tick Drawing

☑ Show Ticks

☑ Show Values

In Value Line Charts however, you can also define the value range, and define at what interval tick marks should be displayed.



Label

Range

● Auto ☑ Include Zero □ Invert Axis

○ Manual Min: Max:

Line

Gridlines

☑ Show Gridlines

Tick Interval

● Auto

○ Manual

Tick Drawing

☑ Show Ticks

☑ Show Values

Axis Position

Left/Bottom At Value / On Category Number: 0

*Label*

The text entered into the `Label` field will be printed below the axis as a description of the X-axis.

*Range*

By default, the `Auto` radio button is selected in the Range group box. If you want to display a fragment of the chart in greater detail, activate the `Manual` radio button and enter minimum and maximum values into the respective fields. If the column that is used for the X-axis does not include zero, you can deactivate the `Include Zero` check box and the X-axis will start with the minimum value that is available in the series. The `Invert Axis` option enables you to invert the values of the X-Axis. For example, if the values run from the **`0`** to **`360`**, selecting this option will generate the X-Axis so that **`360`** is at the origin and the values progress down to **`0`** as the X-Axis goes upwards.

*Line*

The axis is displayed in the color that you choose from the `Line` drop-down-list. You can use one of the preselected colors, or click the **Other color...** button to choose a standard color or define a custom color. Click the **Select...** button on the Custom tab and use the pipette to pick a color that is displayed somewhere on your screen.

*Grid lines*

If the `Show Grid lines` check box is activated, you can choose a color from the corresponding drop-down list box.

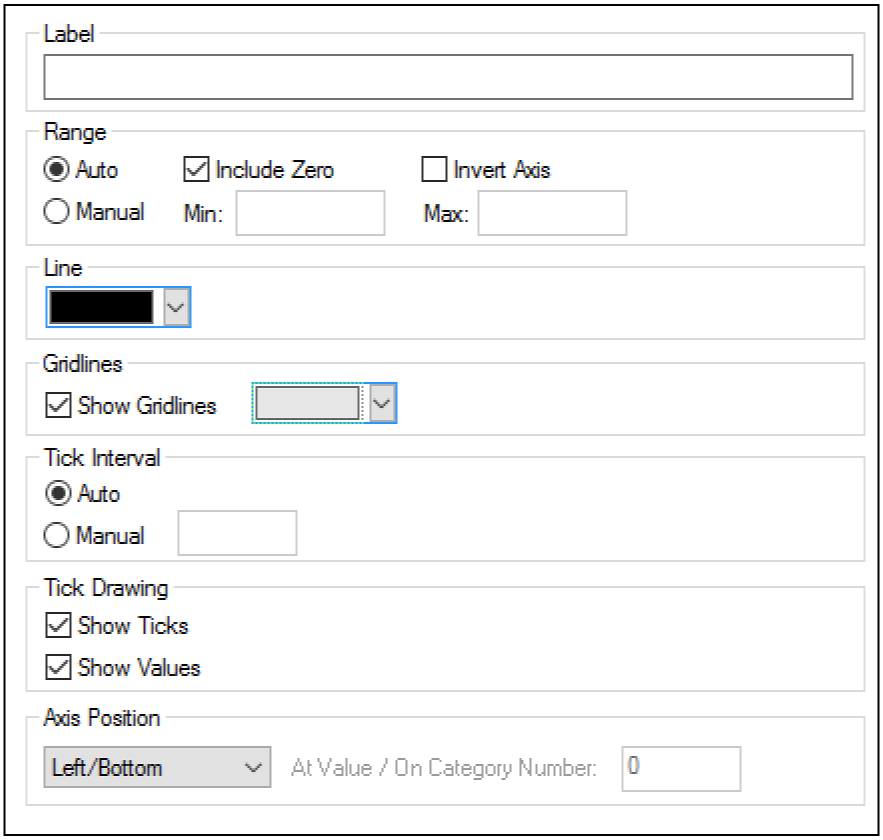
*Tick Interval*
If you are not satisfied with the default tick marks, you can activate the `Manual` radio button in the Tick Interval group box and enter the difference between the individual tick marks into the corresponding field.

*Tick Drawing*
You can switch the display of tick marks on the axis and/or axis values on or off.

*Axis Position*
From the drop-down list, you can choose the position where the axis is to be displayed. When selecting "At Value / On Category Number", you can also position the axis anywhere within the plot.

## 12.9.3.2.5 Y-Axis

In the Y-axis section of the Change Appearance dialog box, you can enter a label for the axis, define colors for the axis and the grid lines (if displayed), define the value range, and decide if and where tick marks should be displayed and whether or not you want to show the axis values. This section is the same for all bar and line charts.



Label

Range
● Auto ☑ Include Zero □ Invert Axis
○ Manual Min: Max:

Line
■

Gridlines
☑ Show Gridlines

Tick Interval
● Auto
○ Manual

Tick Drawing
☑ Show Ticks
☑ Show Values

Axis Position
Left/Bottom At Value / On Category Number: 0

*Label*
The text entered into the `Label` field will be printed to the left of the axis as a description of the Y-axis.

*Range*
By default, the Auto radio button is selected in the Range group box. If you want to display a fragment of the chart in greater detail, activate the Manual radio button and enter minimum and maximum values into the respective fields. If the column that is used for the Y-axis does not include zero, you can deactivate the Include Zero check box and the Y-axis will start with the minimum value that is available in the series. The Invert Axis option enables you to invert the values of the Y-Axis. For example, if the values run from the **0** to **360**, selecting this option will generate the Y-Axis so that **360** is at the origin and the values progress down to **0** as the Y-Axis goes upwards.

*Line*
The axis is displayed in the color that you choose from the Line drop-down-list. You can use one of the preselected colors, or click the **Other color...** button to choose a standard color or define a custom color. Click the **Select...** button on the Custom tab and use the pipette to pick a color that is displayed somewhere on your screen.

*Grid lines*
If the Show Grid lines check box is activated, you can choose a color from the corresponding drop-down list box.

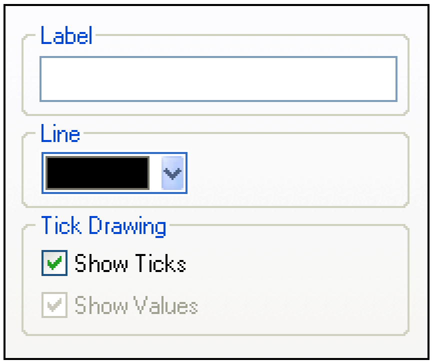
*Tick Interval*
If you are not satisfied with the default tick marks, you can activate the Manual radio button in the Tick Interval group box and enter the difference between the individual tick marks into the corresponding field.

*Tick Drawing*
You can switch the display of tick marks on the axis and/or axis values on or off.

*Axis Position*
From the drop-down list, you can choose the position where the axis is to be displayed. When selecting "At Value / On Category Number", you can also position the axis anywhere within the plot.

### 12.9.3.2.6 Z-Axis

In the Z-axis section of the Change Appearance dialog box, you can enter a label for the axis, define colors for the axis, and decide whether or not you want to show tick marks on the axis. This section is the same for all 3d bar charts (Bar Chart 3d and Bar Chart 3d Grouped).



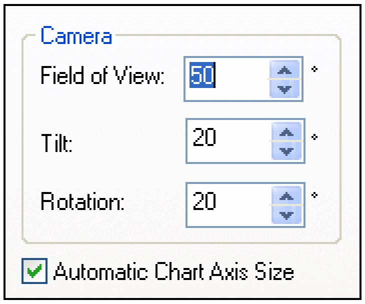
*Label*
The text entered into the `Label` field will be printed to the right of the axis as a description of the Z-axis.

*Line*
The axis is displayed in the color that you choose from the `Line` drop-down-list. You can use one of the preselected colors, or click the **Other color...** button to choose a standard color or define a custom color. Click the **Select...** button on the Custom tab and use the pipette to pick a color that is displayed somewhere on your screen.

*Tick Drawing*
You can switch the display of tick marks on the axis on or off.

### 12.9.3.2.7 3D Angles

In 3d bar charts you can customize the 3d appearance of the chart in the 3d Angles section of the Change Appearance dialog box.



Camera

Field of View: 50 °

Tilt: 20 °

Rotation: 20 °

☑ Automatic Chart Axis Size

The **Field of view** option causes the diagram to appear as if observed from a small or great distance. Values ranging from 1 through 120 are valid. Higher values cause the diagram to appear as if observed from a greater distance.

The **Tilt** value determines the rotation around the X-axis, whereas the **Rotation** value defines the rotation around the Y-axis. You can automatically adapt the size of the chart axis to the Chart window width by selecting the corresponding check box.

If the **Automatic Chart Axis Size** check box is selected, MobileTogether Designer will automatically calculate the optimum size of the X-axis as well as the Y-axis for the current Chart window size. The width and height of the chart will change dynamically when you resize the Chart window.

### 12.9.3.2.8 Sizes

In the Sizes section of the Change Appearance dialog box, you can define different margins as well as the size of axis and gauge ticks. Note that not all the properties listed below are available for all chart types.

*General*
Outside margin Space between the plot and the edge of the Chart window.

| | |
|---|---|
| Title to Plot | Space between the chart title and the upper edge of the plot. |
| Legend to Plot | Space between the lower edge of the plot and the legend. |

*Pie*

| | |
|---|---|
| Plot to Label | In pie charts, the space between the most left and right edge of the pie and its labels. |
| Pie Height | In 3d pie charts, the height of the pie. |
| Pie Drop Shadow | In 3d pie charts, the length of the shadow (if it is activated in the Pie section). |

*X-Axis*

| | |
|---|---|
| X-Axis to Axis Label | In bar and line charts, the space between the X-axis and its label. |
| X-Axis to Plot | In 2d bar charts and line charts, the space between the X-axis and the plot. |
| X-Axis Tick Size | In bar and line charts, the length of the ticks on the X-axis. |

*Y-Axis*

| | |
|---|---|
| Y-Axis to Axis Label | In bar and line charts, the space between the Y-axis and its label. |
| Y-Axis to Plot | In 2d bar and line charts, the space between the Y-axis and the plot. |
| Y-Axis Tick Size | In bar and line charts, the length of the ticks on the Y-axis. |

*Z-Axis*

| | |
|---|---|
| Z-Axis to Axis Label | In 3d bar charts, the space between the Z-axis and its label. |
| Z-Axis Tick Size | In 3d bar charts, the length of the ticks on the Z-axis. |

*Line Drawing*

| | |
|---|---|
| Connection Shape Size | In line charts, the size of the squares that mark the values in the chart. |
| Line width | In line charts, the width of the line. |

*3d Axis Sizes*

| | |
|---|---|
| Manual X-Axis Size of Base | In 3d bar charts, defines the relation between the length of the X-axis and the Chart window size. Please note that the `Automatic Chart Axis Size` check box in the 3d Angles section must be deactivated, otherwise the size will still be calculated automatically. |
| Manual Y-Axis Size of Base | In 3d bar charts, defines the relation between the length of the Y-axis and the Chart window size. Please note that the `Automatic Chart Axis Size` check box in the 3d section must be deactivated, otherwise the size will still be calculated automatically. |
| Z-Axis Series Margin | In 3d bar charts, the distance on the Z-axis between the individual series. |

*Gauge*

| | |
|---|---|
| Border Width | In round gauge charts, the width of the border around the gauge. |

*Gauge Ticks*

| | |
|---|---|
| Border to Tick Distance | In round gauge charts, the space between the inner edge of the border and the ticks that mark the values. |
| Major Tick Length | In round gauge charts, the length of the major ticks (i.e., ticks that show a label). |
| Major Tick Width | In round gauge charts, the width of the major ticks (i.e., ticks that show a label). |
| Minor Tick Length | In round gauge charts, the length of ticks that do not have a value displayed. |
| Minor Tick Width | In round gauge charts, the width of ticks that do not have a value displayed. |

*Gauge Needle*

| | |
|---|---|
| Needle Length | In round gauge charts, the length of the needle. (Note that the percentage is calculated from the diameter of the gauge, so if you choose a value greater than 50%, the needle will point to somewhere outside the gauge!) |
| Needle Width at Base | In round gauge charts, the width of the needle at the center of the gauge. |
| Needle Base Radius | In round gauge charts, the radius of the base that covers the center of the gauge. |

*Gauge Color Range*

| | |
|---|---|
| Border to Color Range Distance | In round gauge charts, the space between the inner edge of the border and the outer edge of the color range[1190]. |
| Color Range Width | In round gauge charts, the width of the customizable color range. (Note that the percentage is calculated from the diameter of the gauge!) |

*Gauge Value*

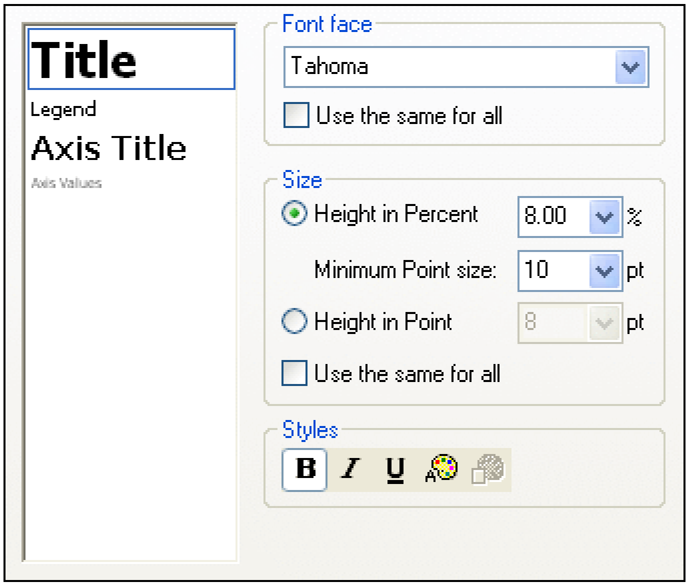
| | |
|---|---|
| Offset to Center | Distance from the center at which the gauge value is displayed. |

*Extra Value*

| | |
|---|---|
| Offset to Center | Distance from the center at which the extra label (defined in the Gauge Chart settings[1184]) is displayed. |

## 12.9.3.2.9 Fonts

The Fonts section of the Change Appearance dialog box lets you configure fonts for objects in the Chart window.



Font face: Tahoma

□ Use the same for all

Size

● Height in Percent 8.00 %

Minimum Point size: 10 pt

○ Height in Point 8 pt

□ Use the same for all

### Font settings

You can choose the font face, size, and style for the individual elements displayed in the Chart window. You can define the size as a percentage of the chart size and define a minimum size in points, or specify an

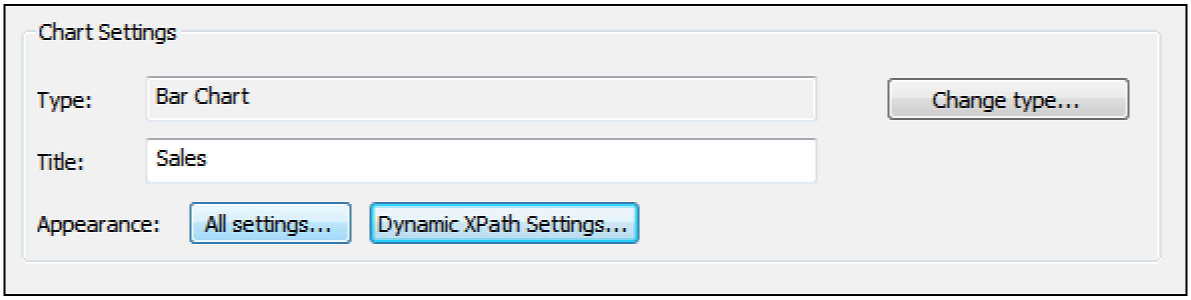
absolute value (in points). To apply the same font and/or size to all text elements, activate the respective `Use the same for all` check box.
The element names in the list box are defined as follows:

- **Title**: The name of a chart
- **Legend**: The key to the colors used in the chart
- **Labels**: The designation of the pieces of a pie chart
- **Axis Title**: The name of the X, Y, and Z axis in a bar or line chart
- **Axis Values**: The units displayed on an axis in a bar or line chart
- **Tick Values**: The units displayed on a gauge chart
- **Values**: The values displayed on the bars of a bar chart

## 12.9.3.3 Dynamic XPath Settings

Dynamic XPath Settings are very useful if you wish to use dynamic data from the XML document in the settings of the chart. For example, the title of a chart about a `Region` element might need to have data about that `Region` element (such as its name) in the title of the chart. If there are several Region elements, any one of which will be used at a time for the chart, then the data for the chart title can only be obtained dynamically via an XPath expression. The Dynamic XPath Settings dialog enables such data to be accessed via XPath expressions.

To access the chart's Dynamic XPath Settings dialog, click **Dynamic XPath Settings** in the Chart Settings pane[1171] of the Chart Configuration dialog (*see screenshot below*).



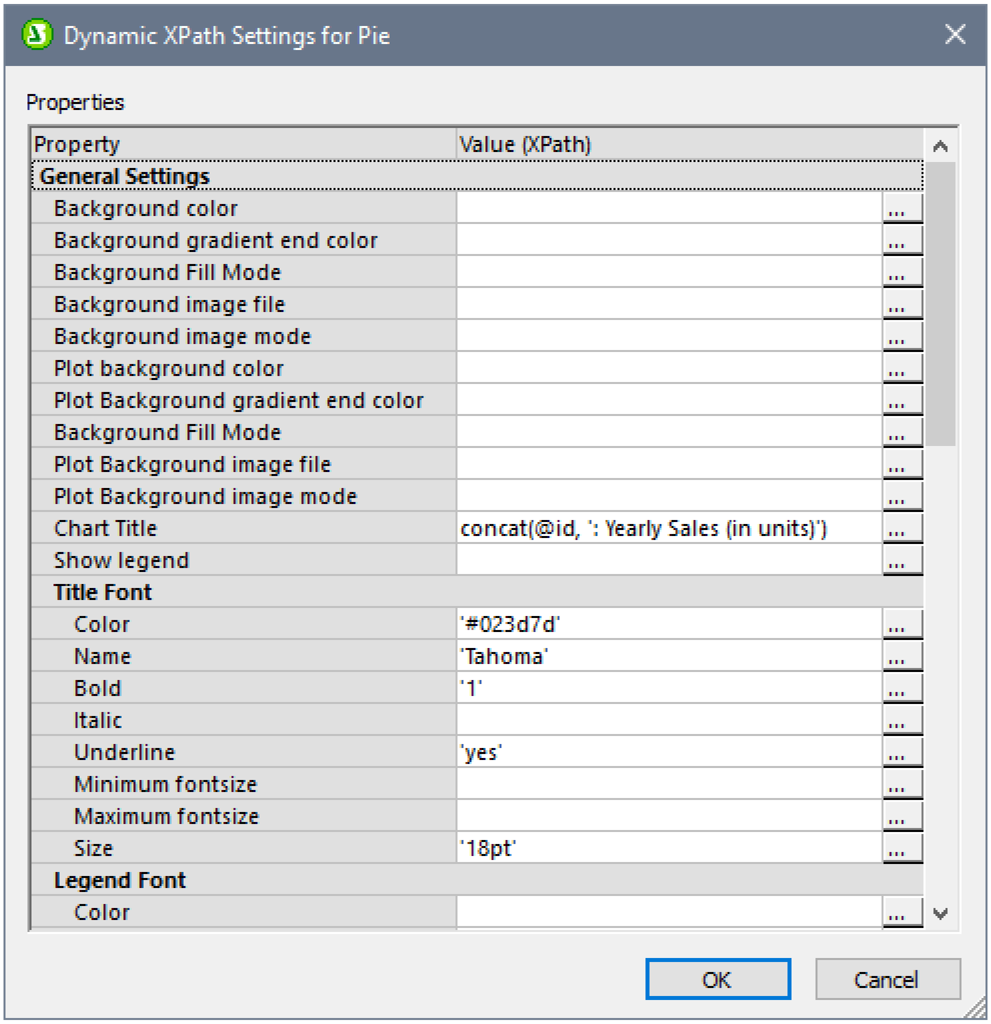
Type: Bar Chart

Title: Sales

Appearance:

### Dynamic XPath Settings

The Dynamic XPath Settings dialog (*screenshot below*) is accessed by clicking the **Dynamic XPath Settings** button in the Chart Settings pane of the Chart Configuration dialog. A number of chart settings can be entered in this dialog.



Dynamic XPath Settings for Pie

Properties

| Property | Value (XPath) |
|---|---|
| **General Settings** | |
| Background color | |
| Background gradient end color | |
| Background Fill Mode | |
| Background image file | |
| Background image mode | |
| Plot background color | |
| Plot Background gradient end color | |
| Background Fill Mode | |
| Plot Background image file | |
| Plot Background image mode | |
| Chart Title | concat(@id, ': Yearly Sales (in units)') |
| Show legend | |
| **Title Font** | |
| Color | '#023d7d' |
| Name | 'Tahoma' |
| Bold | '1' |
| Italic | |
| Underline | 'yes' |
| Minimum fontsize | |
| Maximum fontsize | |
| Size | '18pt' |
| **Legend Font** | |
| Color | |

Note the following points:

- All entries in this dialog are evaluated as XPath expressions, so string literals must be enclosed in quotes. For example: `'Tahoma'`, `'1'`, `'18pt'`, and `'#FF3366'`.
- On hovering over a setting or its value, a tooltip appears giving information about enumerations and formats.
- Dynamic XPath settings have precedence over settings made in the Chart Configuration dialog or Change Appearance dialog[1178]. For example, a chart title made as a dynamic XPath setting has precedence over one set in the Chart Configuration dialog.
- The colors of the color schema will be used when the user-defined color schema is selected in the Change Appearance dialog[1178] as the color schema to use. Colors are set in the RGB hexadecimal format: `#RRGGBB`. So an XPath expression to specify the color red would be: `'#FF0000'`.

# 12.10 Control Templates

A control template is a template that you can reuse at multiple locations in the pages of your design. The main steps and key mechanisms for using control templates are listed briefly below and described in detail in the sub-sections of this section. The Example Projects 1210 section shows, through examples, the different ways in which control templates can be used.

## Creating a control template

A **control template** 1201 is designed like a page. You can add controls to it that structure the design of the page and use data from page sources.

Note the following key features:

- You can create multiple control templates in a project.
- You can reuse a control template at different locations in the design.
- You can change the design of a page by switching templates
- You can select a template based on runtime conditions or user input
- You can modify the contents of a template based on runtime conditions or user input

These uses are demonstrated in the project files described in the Example Projects 1210 section.

## Using a control template via the Placeholder Control

After a control template has been created, you can use it via a **Placeholder Control** 1205. This is a control which you (i) place at the location where you want to use the control template, and (ii) associate with the control template you want to instantiate there.

The Placeholder Control's properties specify the following:

- Which control template to use at the Placeholder Control's location
- The XPath expressions that define the parameter values to pass to the selected control template
- A new context node for XPath expressions that are used in the control template

All of these properties are specified as values of the Placeholder Control's properties in the Styles & Properties Pane 274.

## This section

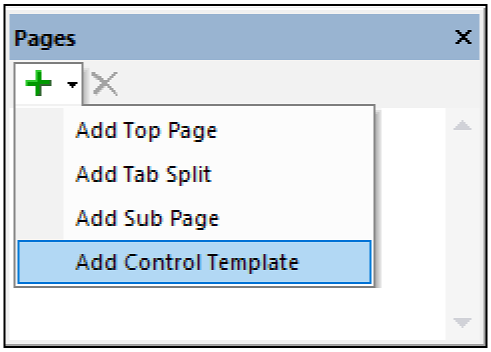
This section consists of the following sub-sections:

- Creating a Control Template 1201
- Using a Control Template: Placeholder Controls 1205
- Overriding Template Events 1208
- Example Projects 1210

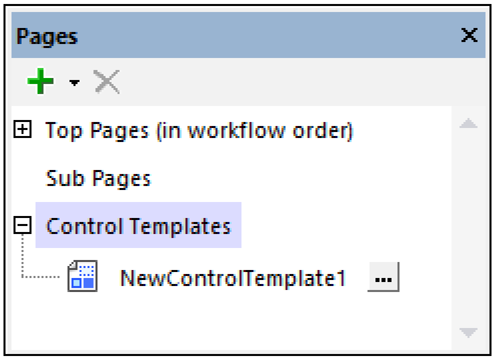
## 12.10.1 Creating a Control Template

Create a control template as follows:

1. In the Pages Pane[257], click the **Add Page** icon and select **Add Control Template** *(see screenshot below)*.



2. Double-click the new control template item that is added to the list *(see screenshot below)* and rename it suitably. The next step will be to declare parameters and define variables for the control template.



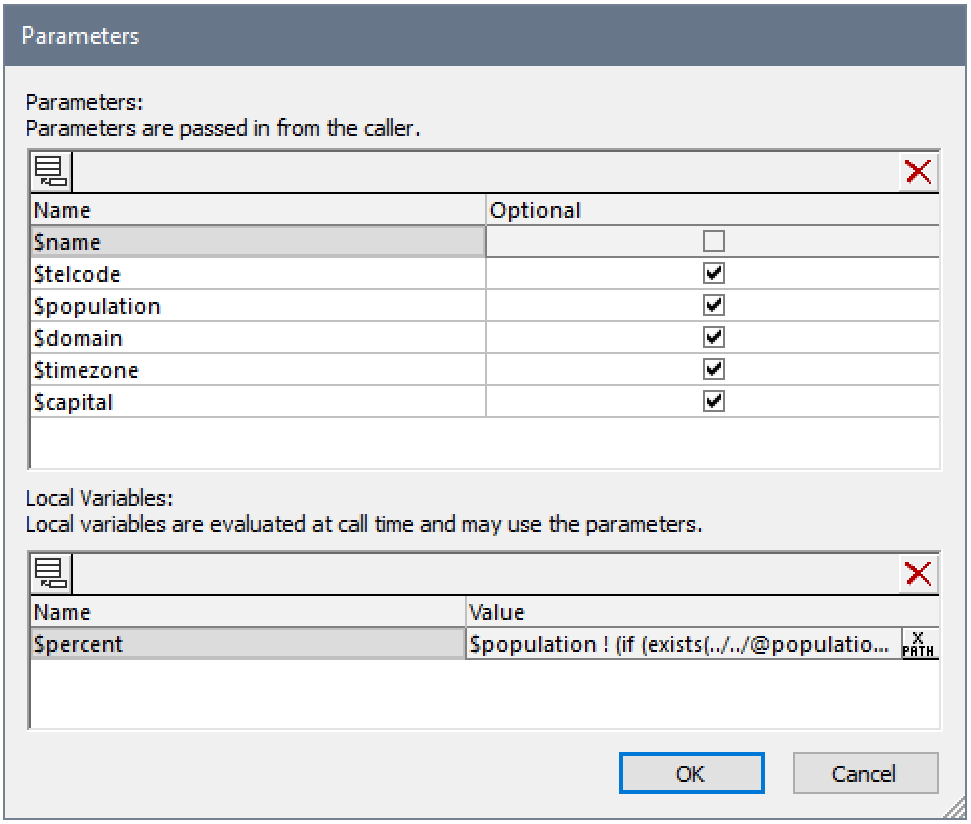
*Create a control template from existing controls*
You can also create a control template quickly if you already have one or more controls on the page that can be converted to a control template.

To create a control template in this way, select the controls you want to convert and right-click. If the controls can be converted, then the **Create Template from Selection** command will be enabled, and you can select it to create a new control template. The selected controls will be placed automatically in the new control template and a Placeholder Control that calls the newly created control template will replace the control/s at their original location. The new control template will be shown in the Pages Pane[257] (*see screenshot above*).

### Parameters and variables

Parameters enable you to use dynamic or static values within the scope of the control template when the template is instantiated. Both, parameters and variables, are declared in the control template. The main difference between the two is that while the values of variables are defined in the control template itself, the values of parameters are defined on the calling page and are passed to the control template's parameters when the control template is instantiated.

If you want to declare parameters and/or variables in a control template, in the Pages Pane[257] *(see screenshot above)*, click the template's **Additional Options** button (which is located to the right of the control template name). In the Parameters dialog that appears *(screenshot below)*, add a parameter or variable (by clicking the **Add Parameter** or **Add Variable** toolbar icon in the top left of the respective pane) and name the parameter/variable suitably (by double-clicking and editing the name).



Parameters:
Parameters are passed in from the caller.

| Name | Optional |
|---|---|
| $name | □ |
| $telcode | ☑ |
| $population | ☑ |
| $domain | ☑ |
| $timezone | ☑ |
| $capital | ☑ |

Local Variables:
Local variables are evaluated at call time and may use the parameters.

| Name | Value |
|---|---|
| $percent | $population ! (if (exists(../../@populatio... |

*Parameters*

Check a parameter's *Optional* box if you want to make the parameter non-mandatory. If a parameter is mandatory, it is an error if it does not receive a value at run time. If, in the calling placeholder control, the value of a parameter is defined by an XPath expression, then this expression is evaluated in the context of the placeholder control's context. The context node is **not** changed by the `Control XPath Context`[568] property of the placeholder control that instantiates the control template[1205].

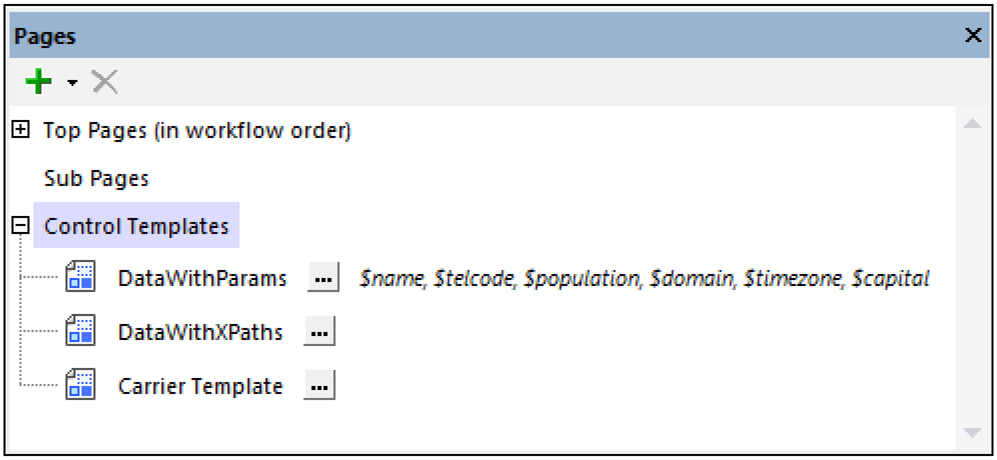
*Variables*

To enter the value of a variable (dynamic or static), double-click in the variable's *Value* field and enter a suitable XPath expression.

Note the following points about how the XPath expressions of parameters and variables are evaluated:

- The context of XPath expressions will be the context node set in the `Control XPath Context`[568] property of the calling placeholder control[1205]. If no value has been set for this property, then the context will be the context node of the calling placeholder control.
- This XPath expression to generate variable values can use the parameters of that control template as well as variables defined earlier in the list of variable definitions. For example: If a control template has parameters **$a**, **$b**, **$c**, and variables **$x**, **$y**, **$z** (in that order), then the variable **$y** can use this list of parameters and variables to generate its value: **$a**, **$b**, **$c**, **$x**, (but not **$z**).

After you have finished declaring parameters and variables for the control template, click **OK**. The parameters you declared are displayed in the Pages Pane[257] as shown for the *DataWithParams* template in the screenshot below. You can edit the parameter/variable list of a control template at any time by clicking the control template's **Additional Options** button to open the Parameters dialog.


Pages
Top Pages (in workflow order)
Sub Pages
Control Templates
DataWithParams
...
$name, $telcode, $population, $domain, $timezone, $capital
DataWithXPaths
...
Carrier Template
...

**Note:** You can change the order of parameters and variables by dragging a parameter/variable and dropping it to a new location.

**Note:** If you change the order of parameters, you need to make sure that the parameter order is correct in the call to the control template (which is made via placeholder controls)[1205]. Do this as follows: (i) Right-click the control template and select **List Usages**; (ii) For each usage (the list is shown in the Listings Pane[281]), check whether the control template is selected via the dropdown list or via an XPath expression; (iii) If the control template is selected via an XPath expression and if its parameters are selected via an array expression (rather than a map expression), then make sure that the order of the parameters in the array is the same as that in the parameter definitions list of the Parameters dialog *(see screenshot further above)*.

## Content of the control template

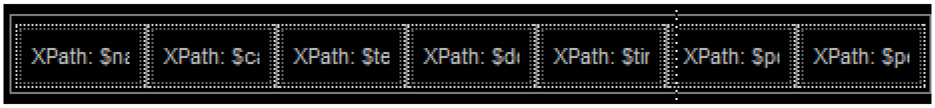
To display the control template's design canvas, select the control template in the Pages Pane[257] *(see screenshot above)*. The design canvas will be displayed in Design View. Here, you can set up and design a set of controls as you do for a normal page. You can use controls, actions, page source nodes, and other components as usual.

Note the following points:

- Only the first visible control in sequential order will be displayed when the control template is instantiated. (A visible control is one that has its `Visible` property set to **`true`**.) This enables you to create multiple controls inside the template and to determine which of these is used by setting XPath conditional expressions on the `Visible` property of the respective controls. The first control with a `Visible` property that evaluates to **`true`** will be used.
- If you wish to display multiple controls when the control template is instantiated, add these controls inside the cells of a Table control[614] with repeating rows or columns—since the table counts as a single control. See screenshot below, where a label has been added to each cell of a single row of the table, and the row itself can repeat. Tables with repeating columns and rows are also used in the

example projects 1210. Note, however, that you cannot add repeating tables 1065 (in which the entire table repeats).

- If you need to use a node from one of the design's page sources and the page source is not shown in the Page Sources Pane 270, then you can add the page source via the pane's **Add Source** icon.
- In the control template's design, you can also use the parameters and variables that you declared for the control template. For example, in the *DataWithParams* control template shown in the screenshot below, a Table control 614 has been created with a single row of seven cells, each of which has a label. Each label has a `Text` value that is generated by an XPath expression that selects a parameter or a variable. The parameter values will be supplied by the calling page and will become the texts of the labels. This control template can be used at different locations and can have different label texts according to the parameter values passed to the template from the calling page.



You can style the controls in a control template in the same way that these controls are usually styled (via their properties 403 and via stylesheets 1318).

After the control template has been created as described above, it can be used in top pages and sub pages 1205, as well as in control templates themselves.

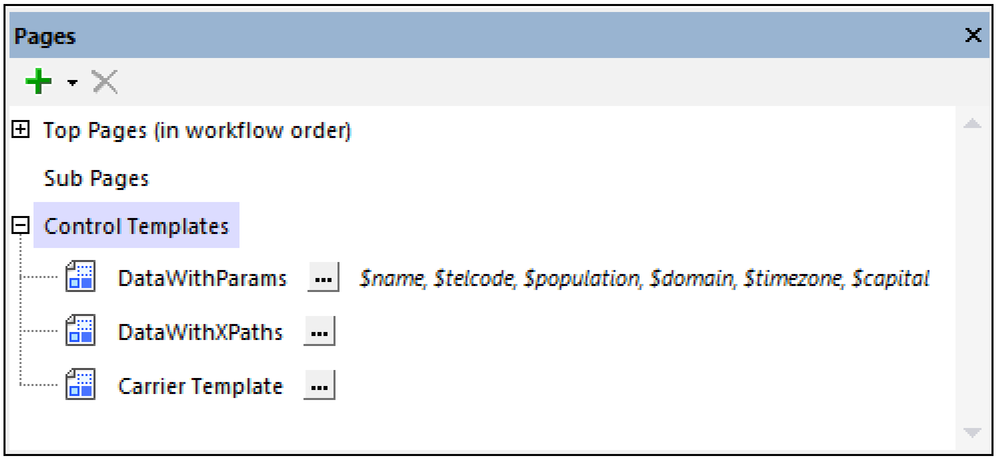
See Example Projects 1210 for examples of how to use control templates.

## The context node of control templates

The context node of a control template will be the context node of the Placeholder Control that calls the control template 1205. If you want to change the context node, you can specify a new context node via the `Control XPath Context` 568 property of the calling placeholder control 1205. You must be aware of what the context node is, since all XPath expressions in the control template must work correctly within this context. If you plan to reuse the control template at various design locations, make sure that the XPath expressions in the control template are correct in each of these different contexts.

## Editing parameters and variables of control templates

You can edit the parameters and variables of a control template (add new, delete selected, or modify properties of) at any time. To do this, go to the control template's Parameters dialog by clicking the **Additional Options** button of the control template. This button is located to the right of the control template name in the Pages Pane 257 *(see screenshot below)*.



Pages
Top Pages (in workflow order)
Sub Pages
Control Templates
DataWithParams
$name, $telcode, $population, $domain, $timezone, $capital
DataWithXPaths
Carrier Template

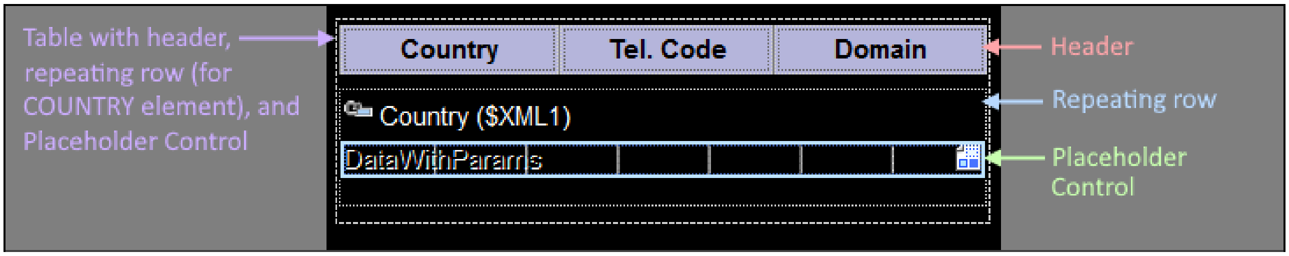
## 12.10.2 Using a Control Template: Placeholder Controls

After you have created a control template[1201], you can use it at one or more locations in page-type components: (i) Top Pages, (ii) Sub Pages, (iii) Control Templates.

To use a control template in a page or control template, do the following:

1. Drop a Placeholder Control[568] in the design, either directly on the page or inside a suitable control such as the Table control[614].
2. Set the Placeholder's **`Control Template`** property to select the control template you want to instantiate at this location.
3. Configure the other settings of the Placeholder Control as required in the Styles & Properties Pane[274].

Inserting a Placeholder Control[568] that calls a control template in a table with repeating elements is especially useful since it enables the template to be applied to each repeating element of the table. In the example shown in the screenshot below, for instance, we have created a table with repeating rows where each of the repeating rows corresponds to a country in an XML dataset that contains multiple sibling **`Country`** elements. Then we dropped a Placeholder Control[568] into the table's repeating row and set its **`Control Template`** property to select the *DataWithParams* template.



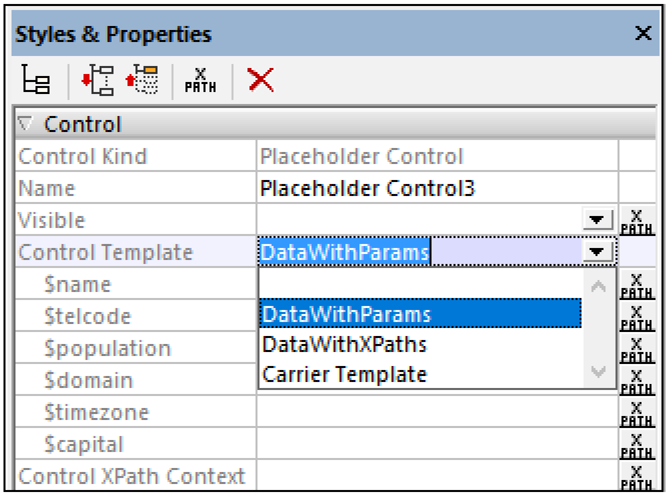
As you can see in the screenshot, the control template has seven cells for seven different country properties. In the current placeholder, three of the properties are instantiated. In another placeholder a different set of properties could be instantiated, enabling the same control template to be called at different locations to generate different sets of data.

Also see the Example Projects[1210] section for more usage examples.

## Configuring the Placeholder Control

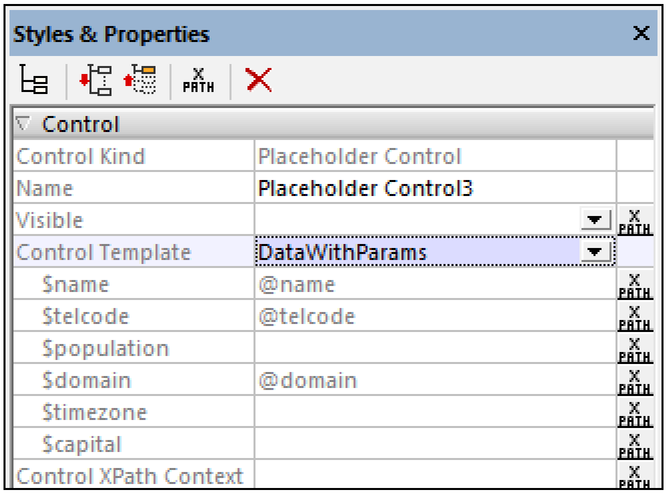
Configure a Placeholder Control as follows:

1. After you have dropped the Placeholder Control[568] in the design, select it in the design.
2. Go to the Styles & Properties Pane[274] and make sure that the Placeholder Control's **`Control Template`** property is correctly set to the name of the control template that you want to insert. (In the screenshot below, the selected control template is *DataWithParams*; it is selected from a dropdown that lists all the control templates in the design.) Alternatively, you can use an XPath expression to select a control template. Such an XPath expression must evaluate to the name of a control template in the design. Using XPath expressions enables you to select templates conditionally. See Example Projects[1210] for ways to do this.



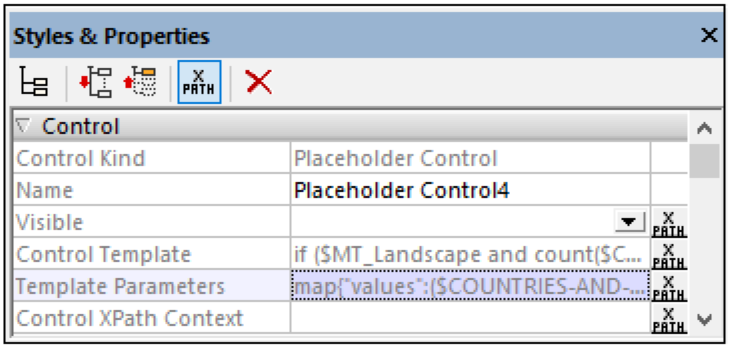
| Control | |
|---|---|
| Control Kind | Placeholder Control |
| Name | Placeholder Control3 |
| Visible | |
| Control Template | DataWithParams |
| $name | |
| $telcode | |
| $population | |
| $domain | |
| $timezone | |
| $capital | |
| Control XPath Context | |

3. After you select the control template, its parameters are displayed as sub-properties of the template *(see screenshot below)*. Enter values for these parameters as XPath expressions. Values must be defined for all mandatory parameters (those that have note been defined as optional[1201]). The context node for evaluating XPath expressions of parameter values is the context node of the Placeholder Control. This context for parameters is not changed by the `Control XPath Context` property *(see below)*.



| Control | |
|---|---|
| Control Kind | Placeholder Control |
| Name | Placeholder Control3 |
| Visible | |
| Control Template | DataWithParams |
| $name | @name |
| $telcode | @telcode |
| $population | |
| $domain | @domain |
| $timezone | |
| $capital | |
| Control XPath Context | |

If you select a control template via an XPath expression (rather than by selecting a name in the property's combo box), then the control template is not selected till runtime, when the expression is evaluated. Since the control template is not known at design time, no parameters can be displayed in the pane. Instead of a list of parameters, a property named `Template Parameters` is available *(see screenshot below)*. You can enter an XPath expression to generate the values of the expected parameters. The expression must be either an array expression or map expression. If you use an array expression, the parameter values must be provided in the same sequence as the parameter definition order in the control template; additionally any optional parameters must not be omitted. (In the case of maps, the keys enable the values to be correctly assigned.) See Example Projects[1221] for a sample of such expressions.



| Control | |
|---|---|
| Control Kind | Placeholder Control |
| Name | Placeholder Control4 |
| Visible | |
| Control Template | if ($MT_Landscape and count($C... |
| Template Parameters | map{"values":($COUNTRIES-AND-... |
| Control XPath Context | |

4. The context node of all XPath expressions inside a control template will by default be the context node of the placeholder control that instantiates (or calls) the template. You can change the context node that is passed to the control template by entering a new context node as the value of the `Control XPath Context` property *(see screenshot above)*. The new context node will be used to evaluate XPath expressions within the template, including expressions that define the values of template variables.

## Convert a Placeholder Control to its template contents

To replace a Placeholder Control with the content of the control template it represents, right-click the placeholder and select the command **Replace Placeholder with Template Content**. Note that neither the control template nor its content is deleted.

## 12.10.3 Overriding Template Events

This is an advanced feature that enables you to specify, on each Placeholder Control[1205], a set of actions to execute when the placeholder's control template is executed. This set of actions overrides the actions defined in the control template, and is triggered when any action event defined in the control template occurs. It is therefore possible to define a different set of overriding actions for each placeholder control. In this way a control template can be modified at the placeholder level, separately for each placeholder control. For example, if three placeholder controls use a single control template, and you want to modify the control template when it is instantiated by one of the three placeholder controls, then you can define a set of overriding actions on this particular placeholder control.

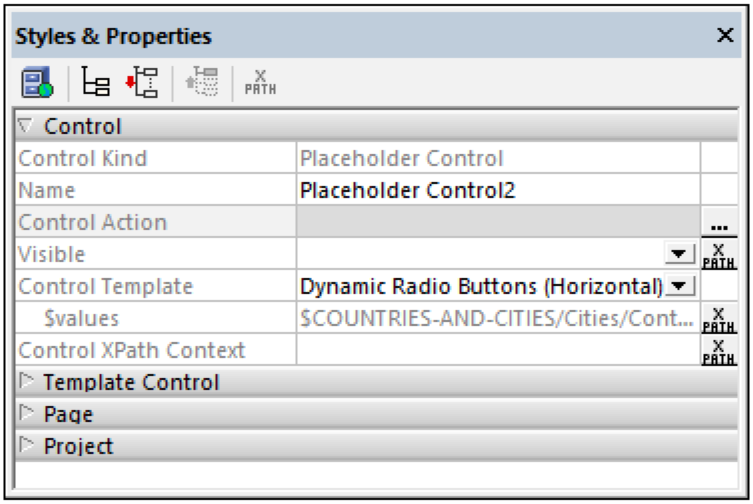
**Note:** If there are multiple placeholder ancestors of a control template, then, when the control template is triggered, the outermost ancestor placeholder is instantiated with any overrides it may have and then calls the next placeholder in the hierarchy. This continues till, eventually, the control template is reached.

**Note:** If you want to prevent a control template being overridden by a placeholder that has an override defined for it, then when the control template is triggered, the first ancestor placeholder that has an override defined for it will be executed.

### How to override control-template actions

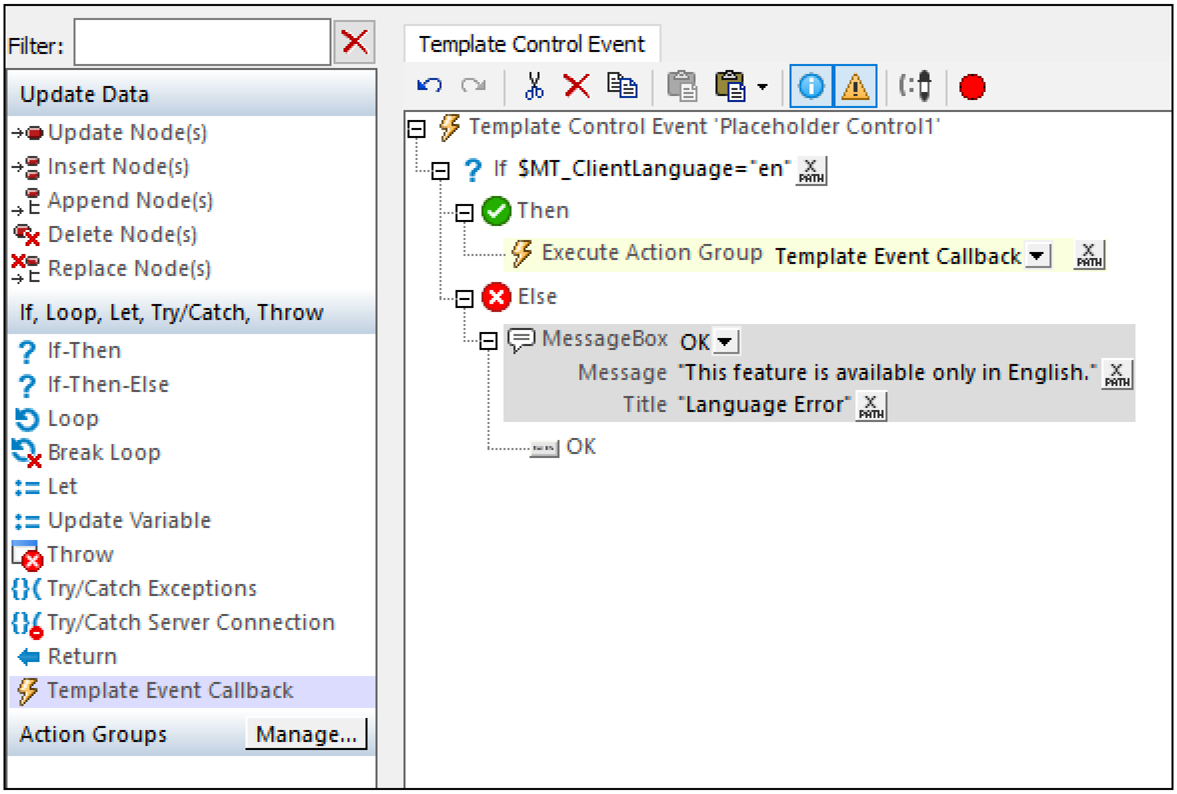
On a Placeholder Control[1205], you can define a set of actions that override the control actions defined in a control template. Do this as follows:

1. Select the Placeholder Control[1205] for which you want to modify control-template actions.
2. In the Styles & Properties Pane[274], go to the placeholder control's properties and click the **Additional Options** button of its Control Action property *(screenshot below)*. Alternatively, right-click the placeholder control and, in the context menu that appears, select **Template Control Event Overrides**.



| | |
|---|---|
| Control Kind | Placeholder Control |
| Name | Placeholder Control2 |
| Control Action | |
| Visible | |
| Control Template | Dynamic Radio Buttons (Horizontal) |
| $values | $COUNTRIES-AND-CITIES/Cities/Cont... |
| Control XPath Context | |

3. In the Actions window that appears, enter the set of actions you want in the Template Control Event pane *(see screenshot below)*.



Template Control Event 'Placeholder Control1'
If $MT_ClientLanguage="en"
Then
Execute Action Group Template Event Callback
Else
MessageBox OK
Message "This feature is available only in English."
Title "Language Error"
OK

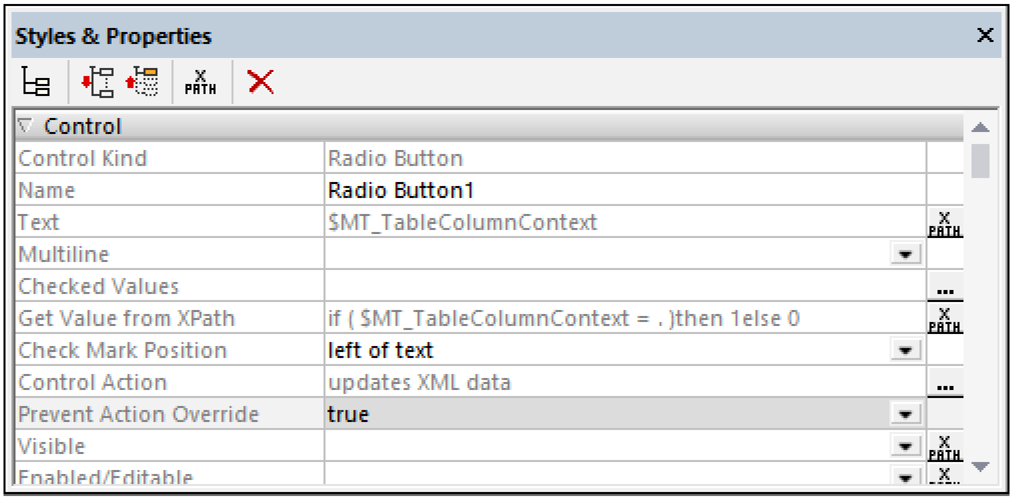
In the screenshot above, the placeholder control's actions have been modified by means of an If-Then-Else 894 action, as follows:

- If the language of the client device is English, then the control template is executed as usual. This is achieved by adding, to the *Then* branch, a Template Event Callback action (shown highlighted in the *Actions* pane at left in the screenshot above). This action simply instantiates the control template.
- If the client language is not English, then a message box is displayed. The Message Box 679 action is added to the *Else* branch.

In the event that the control template contains more than one control, you can select different actions for each control event by using conditional processing that tests the value of the **`$MT_ControlKind`** 1304 or **`$MT_ControlName`** 1304 variables.

## Preventing overrides

If you want to prevent the actions of a control in a control template from being overridden by a placeholder's actions, then set the control's `Prevent Action Override` property to **`true`** (*see screenshot below*). This property is also available for placeholders in a control template.



Styles & Properties

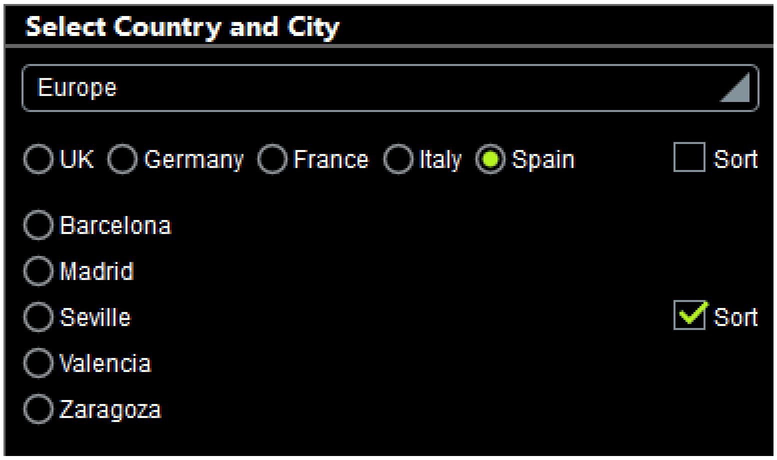
▽ Control

| | |
|---|---|
| Control Kind | Radio Button |
| Name | Radio Button1 |
| Text | $MT_TableColumnContext |
| Multiline | |
| Checked Values | |
| Get Value from XPath | if ( $MT_TableColumnContext = . )then 1else 0 |
| Check Mark Position | left of text |
| Control Action | updates XML data |
| Prevent Action Override | true |
| Visible | |
| Enabled/Editable | |

Note that the `Prevent Action Override` property is available only for controls and placeholders in control templates. Its default value is **`false`**, and it determines whether any overrides defined on placeholders that use the control template will be overridden or not.

## 12.10.4 Example Projects

There are four example projects in this section, each of which builds on the previous project. The first project shows how to create a control template and use it. The goal is a simple one: to let the user select a continent, then a country, and a city *(see screenshot below)*. A selection at each level generates a list of radio buttons at the next level. We reuse control templates to generate these radio button lists. Subsequent projects add functionality and demonstrate different ways in which control templates can be used.



Select Country and City

Europe

○ UK ○ Germany ○ France ○ Italy ● Spain □ Sort

○ Barcelona
○ Madrid
○ Seville ☑ Sort
○ Valencia
○ Zaragoza

To open and experiment with any of the example projects, simply double-click its MTD file *(see below for location)*.

## Location of project files

All the example project files listed below are located in the *(My) Documents* folder: `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\ControlTemplates`.

- **`Cities1-Reuse.mtd`**
- **`Cities2-SwitchTemplates.mtd`**
- **`Cities3-DynamicUpdates.mtd`**
- **`Cities4-DynamicSelection.mtd`**
- **`CitiesWorldwide.xml`**

*Listing of CitiesWorldwide.xml*

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Cities in which the survey is being carried out -->
<Cities>
   <Continent name="Americas">
      <Country name="USA">
         <City name="Miami"/>
         <City name="San Francisco"/>
         <City name="Chicago"/>
         <City name="Washington"/>
         <City name="Los Angeles"/>
         <City name="Boston"/>
         <City name="New Orleans"/>
         <City name="Denver"/>
         <City name="Seattle"/>
         <City name="New York"/>
         <City name="Philadelphia"/>
         <City name="Minneapolis"/>
      </Country>
      <Country name="Canada">
         <City name="Vancouver"/>
         <City name="Ottawa"/>
         <City name="Toronto"/>
         <City name="Montreal"/>
         <City name="Calgary"/>
         <City name="Edmonton"/>
      </Country>
      <Country name="Mexico">
         <City name="Mexico City"/>
         <City name="Guadalajara"/>
         <City name="Puebla"/>
         <City name="Tijuana"/>
         <City name="Monterrey"/>
      </Country>
   </Continent>
   <Continent name="Europe">
      <Country name="UK">
         <City name="London"/>
         <City name="Edinburgh"/>
         <City name="Cambridge"/>
         <City name="Oxford"/>
         <City name="Liverpool"/>
         <City name="Bristol"/>
      </Country>
      <Country name="Germany">
         <City name="Heidelberg"/>
         <City name="Stuttgart"/>
         <City name="Berlin"/>
         <City name="Frankfurt"/>
         <City name="Hamburg"/>
         <City name="Bonn"/>
         <City name="Munich"/>
      </Country>
      <Country name="France">
         <City name="Paris"/>
         <City name="Marseille"/>
         <City name="Lyon"/>
         <City name="Toulouse"/>
         <City name="Nice"/>
         <City name="Strasbourg"/>
      </Country>
      <Country name="Italy">
         <City name="Rome"/>
         <City name="Milan"/>
         <City name="Naples"/>
         <City name="Turin"/>
         <City name="Palermo"/>
         <City name="Genoa"/>
         <City name="Bologna"/>
         <City name="Florence"/>
         <City name="Venice"/>
      </Country>
      <Country name="Spain">
         <City name="Madrid"/>
         <City name="Zaragoza"/>
         <City name="Barcelona"/>
         <City name="Valencia"/>
         <City name="Seville"/>
      </Country>
   </Continent>
   <Continent name="Asia">
      <Country name="Japan">
         <City name="Tokyo"/>
         <City name="Osaka"/>
         <City name="Nagoya"/>
         <City name="Sendai"/>
         <City name="Sapporo"/>
      </Country>
      <Country name="China">
         <City name="Beijing"/>
         <City name="Shanghai"/>
         <City name="Tianjin"/>
         <City name="Hong Kong"/>
      </Country>
      <Country name="India">
         <City name="New Delhi"/>
         <City name="Mumbai"/>
```

```
            <City name="Chennai"/>
            <City name="Bengaluru"/>
            <City name="Kolkata"/>
         </Country>
      </Continent>
      <Continent name="Oceania">
         <Country name="New Zealand">
            <City name="Auckland"/>
            <City name="Wellington"/>
            <City name="Christchurch"/>
         </Country>
         <Country name="Australia">
            <City name="Canberra"/>
            <City name="Melbourne"/>
            <City name="Brisbane"/>
            <City name="Sydney"/>
            <City name="Adelaide"/>
            <City name="Hobart"/>
            <City name="Perth"/>
         </Country>
      </Continent>
      <Continent name="Africa">
         <Country name="South Africa">
            <City name="Pretoria"/>
            <City name="Johannesburg"/>
            <City name="Cape Town"/>
            <City name="Port Elizabeth"/>
            <City name="Durban"/>
         </Country>
         <Country name="Egypt">
            <City name="Cairo"/>
            <City name="Alexandria"/>
         </Country>
         <Country name="Nigeria">
            <City name="Lagos"/>
            <City name="Abuja"/>
            <City name="Kano"/>
         </Country>
         <Country name="Kenya">
            <City name="Nairobi"/>
            <City name="Mombasa"/>
         </Country>
      </Continent>
   </Cities>
```

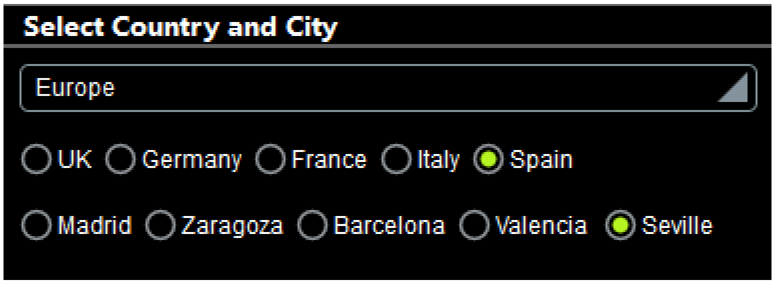
## In this section

This section consists of the following topics:

- Reusing a Template 1214
- Switching Templates 1217
- Dynamically Changing Template Content 1219
- Dynamically Selecting a Template 1221

### 12.10.4.1 Reusing a Template

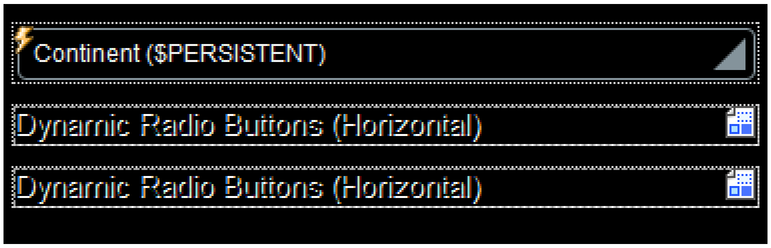
One of the main benefits of control templates is reuse. After you have created a control template, you can use it in different locations in your design. In our example `Cities1-Reuse.mtd`, the goal is to let users select their city by choosing continent, country, and city in that order as shown in the screenshot below. (All the geographic data is referenced from the XML page source `CitiesWorldwide.xml`.)

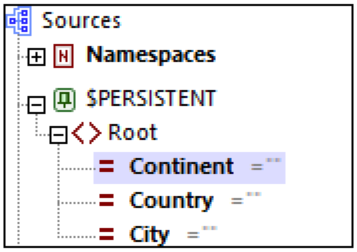

Select Country and City
Europe
○ UK ○ Germany ○ France ○ Italy ● Spain
○ Madrid ○ Zaragoza ○ Barcelona ○ Valencia ● Seville

Since we are using two sets of horizontal radio buttons (one for countries and one for cities), we have created a single control template, named *Dynamic Radio Buttons (Horizontal)*, and used it twice: once to generate radio buttons of the countries of the selected continent, and a second time to generate radio buttons of the cities of the selected country.

#### The project design

The project consists of the control template *Dynamic Radio Buttons (Horizontal)*, which is used twice in the top page named *Select Country and City*. The design of the top page is shown in the screenshot below left. After the user selects a continent in the combo box, radio buttons of the countries in that continent are displayed. After a country is selected, radio buttons of the cities in that country are displayed.


Continent ($PERSISTENT)
Dynamic Radio Buttons (Horizontal)
Dynamic Radio Buttons (Horizontal)



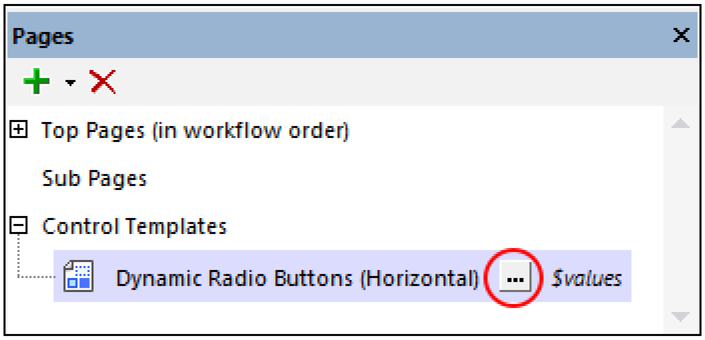
The continent, country, and city selected by the user are each stored in separate nodes of the `$PERSISTENT` page source *(see screenshot above right)*. Dropping these three page source nodes (the **`@Continent`**, **`@Country`**, and **`@City`** attributes of the **`$PERSISTENT/Root`** element) on their respective design components performs certain important functions:

- **`@Continent`** on the combo box: updates **`@Continent`**, with the selection in the combo box and makes **`@Continent`** the context node of the combo box.
- **`@Country`** on placeholder-control-1 makes **`@Country`** the context node of the placeholder control (and of the control template that the placeholder control calls). The value in **`@Country`** is updated via the *OnFinishEditing* action of the radio button in the control template *(see below)*.
- **`@City`** on placeholder-control-2 makes **`@City`** the context node of the placeholder control (and of the control template that the placeholder control calls). The value in **`@City`** is updated via the *OnFinishEditing* action of the radio button in the control template *(see below)*.

The setting of a context node for the placeholder control is important because all XPath expressions in the control template called by the placeholder control will be evaluated relative to this context node.

## The control template "Dynamic Radio Buttons (Horizontal)"

The control template has been declared to have one mandatory parameter named **`$values`**. You can see this declaration by clicking the **Additional Components** button in the Pages Pane *(circled in red in the screenshot below)*.



Pages

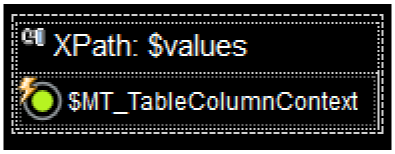
⊞ Top Pages (in workflow order)

Sub Pages

⊟ Control Templates

Dynamic Radio Buttons (Horizontal) ... *$values*

The design of the control template is shown in the screenshot below. The control template consists of a table that expands horizontally, that is, a new column is created for each item of the sequence over which the table expands. This sequence (for the expansion of the table) is defined to be that held in the **`$values`** parameter (as shown in the screenshot below). Each column contains a radio button. So, if **`$values`** holds a sequence of country names, then a column containing a radio button will be created for each country name in the sequence. And if **`$values`** holds a sequence of city names, then a radio button will be created for each city name in the sequence. How the contents of **`$values`** is defined is explained below.



XPath: $values

$MT_TableColumnContext

The radio button control has the following settings (defined in the Styles & Properties Pane[274]):

- A `Text` property that determines the text that accompanies each radio button. The text in our example is defined to be the contents of the **`$MT_TableColumnContext`** variable. This variable is a MobileTogether application variable that has been specially devised for use with tables that expand horizontally. It contains, for each column of such tables, the current context node. So, in our example, if a new column is created due to a country name in the **`$values`** sequence, then that country name is the context node for that column and that country name will be stored in **`$MT_TableColumnContext`** variable while that column is being generated. Since the `Text` property of the radio button has been set to **`$MT_TableColumnContext`**, each radio button will have the name of the sequence item that caused the generation of the current column.
- The `Control Action` property (for *OnFinishEditing*) updates the current node with the value of **`$MT_TableColumnContext`**. So: firstly, what is the current node? It is the context node of the calling placeholder control *(see above)*. In our example, it is, respectively (for the first and second placeholder), the **`@Country`** and **`@City`** attributes of the **`$PERSISTENT/Root`** element. Secondly, which **`$MT_TableColumnContext`** value (that is, of which column) will be passed to the current node? Answer:

The column containing the radio button that was clicked. Consequent to the above: The country/city name of the clicked radio button will be passed to the `@Country` or `@City` attribute, respectively.

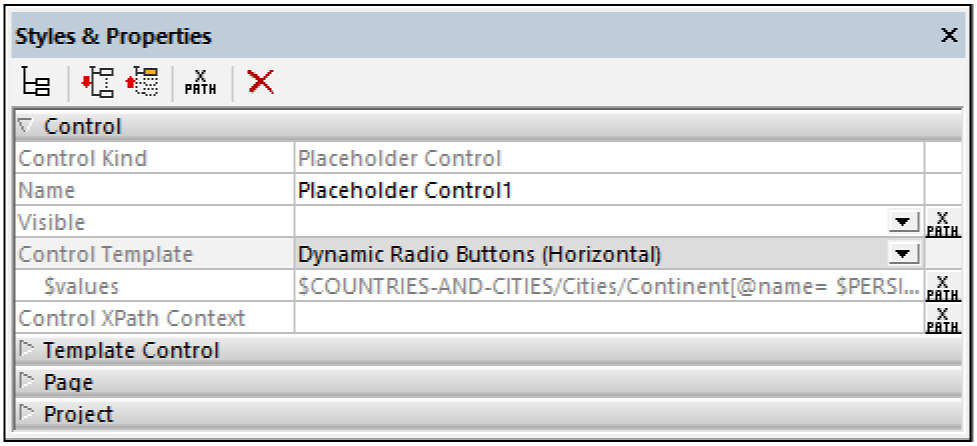
- The `Checked Value` (or `Get Value from XPath`) property of the radio button defines whether the button has been selected (a value of `1`) or not (a value of `0`). The XPath expression that determines this value in the control template is: `if ($MT_TableColumnContext = .) then 1 else 0`. This expression sets the value of that radio button to `1` that has a `$MT_TableColumnContext` value that is equal to the value of the current node (which is the context node of the placeholder control and, in our example, either the `@Country` or `@City` attribute of the `$PERSISTENT/Root` element). Since the current node was updated with the value of `$MT_TableColumnContext` via the *OnFinishEditing* action, only the radio button that was clicked (and triggered the *OnFinishEditing* action) will have a value of `1`.

All that is needed now is to specify the sequence that is passed to the control template's `$values` parameter. This definition is set on the placeholder control *(see below)*.

**Note:** The *Dynamic Radio Buttons (Horizontal)* enables you to create horizontal radio buttons for **any** sequence of values that is supplied to the `$values` parameter.

## The two Placeholder Controls that reuse the control template

A Placeholder Control[568] is created for a control template by dropping the Placeholder Control in the design at the required location, and setting its `Control Template` property to select the control template you want. Alternatively, you can drop the wanted control template in the design at the required location; the `Control Template` property will be set automatically.



| Control | | |
|---|---|---|
| Control Kind | Placeholder Control | |
| Name | Placeholder Control1 | |
| Visible | | X PATH |
| Control Template | Dynamic Radio Buttons (Horizontal) | |
| $values | $COUNTRIES-AND-CITIES/Cities/Continent[@name= $PERSI... | X PATH |
| Control XPath Context | | X PATH |
| Template Control | | |
| Page | | |
| Project | | |

The key settings for the Placeholder Control in our example are explained below. The screenshot above shows the properties of the first Placeholder Control, which is used to select countries.

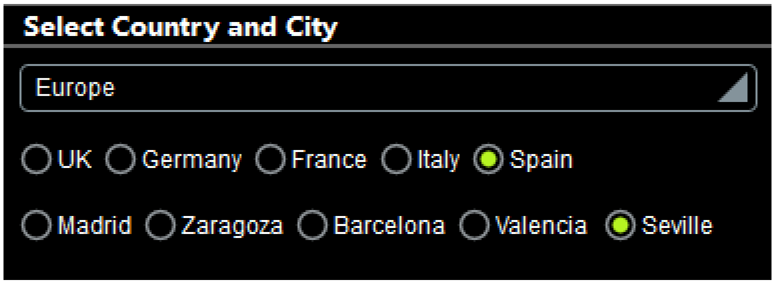
- *Associated Page Source Node:* This specifies the page source node that will be the context node of the Placeholder Control, and, by extension, the context node of the control template that is called. All XPath expressions in the control template (including for its parameters and variables) will be evaluated relative to this node. You can make this setting by dropping the page source node onto the Placeholder Control[568]. In our project the associated page source nodes for the two Placeholder Controls[568] are, respectively, the `@Country` and `@City` attributes of the `$PERSISTENT/Root` element.
- *Control Template:* Selects the control template that will be instantiated at the location of the Placeholder Control[568]. You can select the control template in the property's combo box. (Alternatively, you can drag a control template into the design and this property will be set automatically to select that control template.)

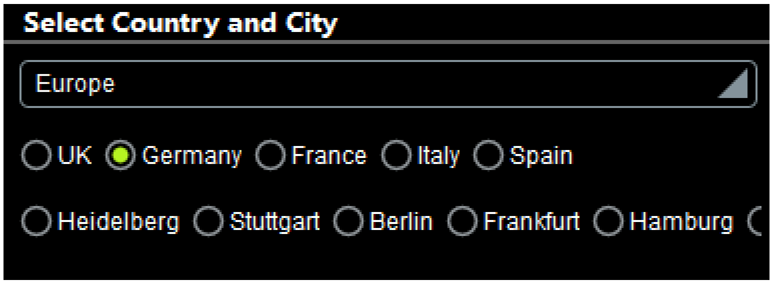
- *Parameters:* Defines the values to be sent to the **`$values`** parameter of the control template. In the case of the Placeholder Control that instantiates radio buttons for the countries of the user-selected continent, the XPath expression is: `$COUNTRIES-AND-CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent]/Country/@name`. The predicate filter in this expression (highlighted in yellow) selects the continent that the user has chosen (and which is now stored in the **`$PERSISTENT`** tree). The XPath expression of the second Placeholder Control, which generates radio buttons for cities of the selected country, is: `$COUNTRIES-AND-CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent]/Country[@name=$PERSISTENT/Root/@Country]/City/@name`.

Notice that the two placeholders have different context nodes and send different sequences to the **`$values`** parameter. This enables the control template to generate the appropriate radio buttons in the two locations where it is instantiated by the respective Placeholder Controls.

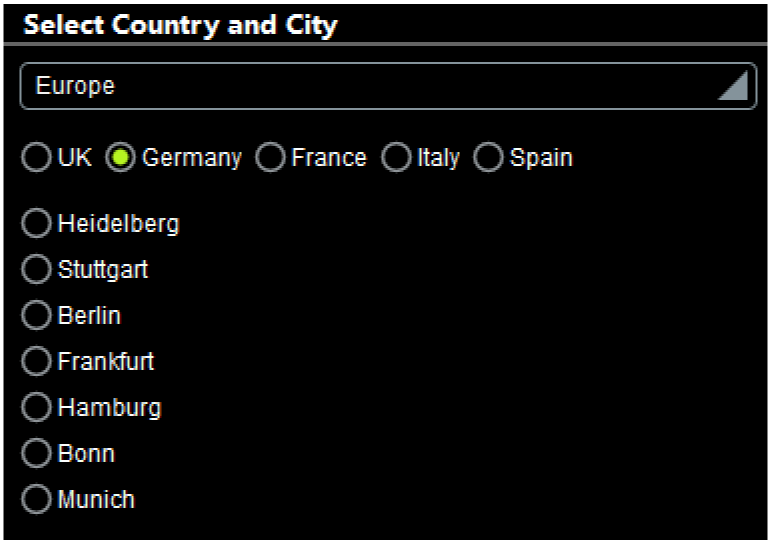
## 12.10.4.2 Switching Templates

A second benefit of control templates is the ability to easily change templates in the Placeholder Control, for example, if you want to change the layout of a component. In the example `Cities1-Reuse.mtd` (described in the previous topic[1214]), we run into the following problem: While the list of cities in some cases is short enough to fit in the horizontal span of the display (such as for Spain), some lists are too long to fit (for example, Germany); *see screenshots below.*


Select Country and City
Europe
○ UK ○ Germany ○ France ○ Italy ● Spain
○ Madrid ○ Zaragoza ○ Barcelona ○ Valencia ● Seville


Select Country and City
Europe
○ UK ● Germany ○ France ○ Italy ○ Spain
○ Heidelberg ○ Stuttgart ○ Berlin ○ Frankfurt ○ Hamburg

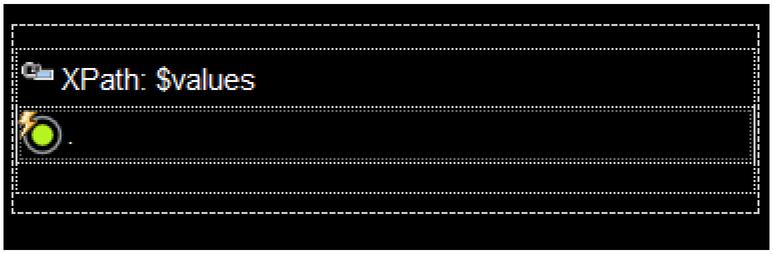
In order to fit all cities in the display, it would be more sensible to display cities in a vertical list. In the example `Cities2-SwitchTemplates.mtd`, we have added (to the previous design) a new control template that generates vertical radio buttons, and in the placeholder control that instantiates the radio buttons for cities, we have simply switched the control template to call the new control template. A simulation of the modified design is shown in the screenshot below.



## The new control template: "Dynamic Radio Buttons (Vertical)"

The control template has been declared to have one mandatory parameter named **`$values`** (just like the control template for horizontal radio buttons). Additionally, however, a variable named **`$source`** has been defined with a value that is the current node.

The design of the control template is shown in the screenshot below. The major difference from the template for horizontal radio buttons is that the table does not expand horizontally (creating a new column for each item in **`$values`**). Instead, a new row is generated for each item in **`$values`**. Since each row contains a radio button, the radio button list is vertical.



The radio button control has the following settings (defined in the Styles & Properties Pane[274]):

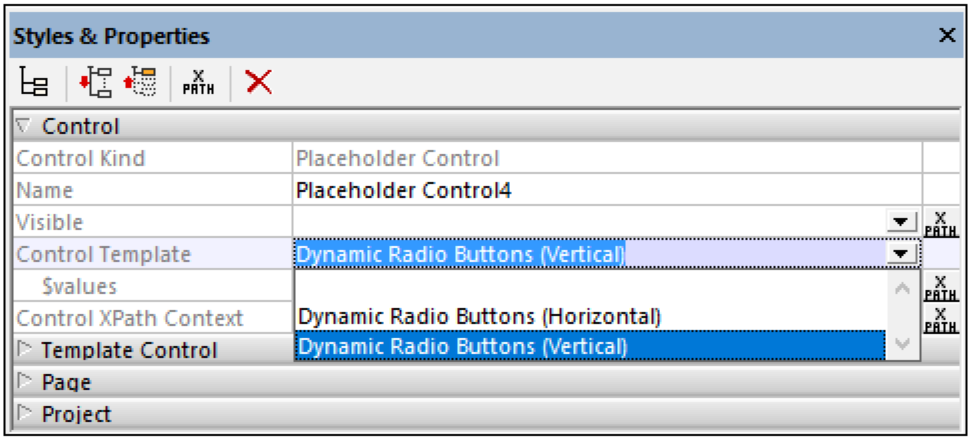
- A `Text` property that selects the current node. Since the current node inside a table row will be the current **`$values`** item for which the row is being generated, the text of each radio button will be the current **`$values`** item.
- The `Control Action` property (for *OnFinishEditing*) updates the **`$source`** variable with the value of the current node (which is the current **`$values`** item). As a result, the **`$source`** variable will contain the text value of the clicked radio button.
- The `Checked Value` (or `Get Value from XPath`) property of the radio button defines whether the button has been selected (a value of **`1`**) or not (a value of **`0`**). The XPath expression that determines this property value is: `if ($source = .) then 1 else 0`. This expression sets the value of that radio button to **`1`** that has a current node value which is equal to the value of **`$source`**. This will be true only

for the clicked radio button, since $source will be updated with the current node value only when the *OnFinishEditing* action is triggered *(see previous point)*.

You now have a new control template that generates vertical radio buttons for the items in the template's **`$values`** parameter. Since this is the same parameter input that the *Dynamic Radio Buttons (Horizontal)* control template takes, you can now set a Placeholder Control to call either the template for horizontal radio buttons or that for vertical radio buttons.

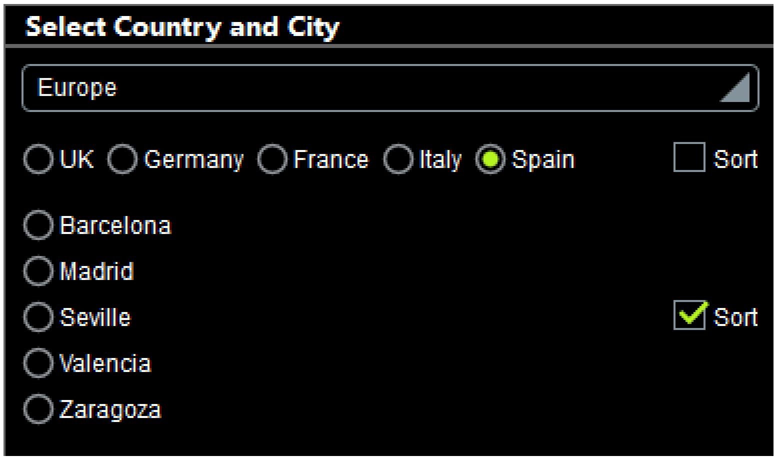
### Switching between templates in the Placeholder Control

In the example file, we go to the top page and, in the design, select the placeholder that instantiates the radio buttons for cities. In the Styles & Properties Pane[274], we can now switch between *Dynamic Radio Buttons (Vertical)* for vertically listed radio buttons and *Dynamic Radio Buttons (Horizontal)* for horizontal radio buttons *(see screenshot below)*. In our example, we have chosen to display cities in a vertical list.



## 12.10.4.3 Dynamically Changing Template Content

Another benefit of using control templates is that you can pass parameter values to the control template and dynamically change content inside the template. In our example file `Cities3-DynamicUpdates.mtd`, we have added sort functionality to the previous example, `Cities2-SwitchTemplates.mtd`. It enables users to decide whether the display of countries and/or cities should be sorted *(see screenshot below)*.


Select Country and City

Europe

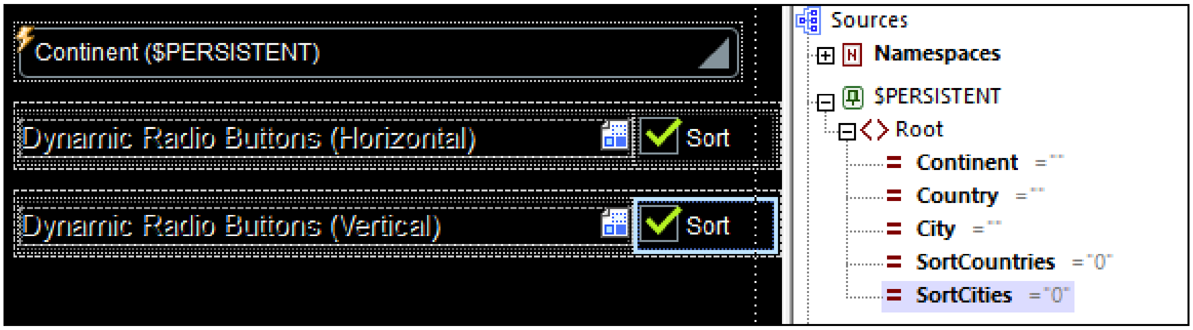
○ UK ○ Germany ○ France ○ Italy ● Spain □ Sort

○ Barcelona
○ Madrid
○ Seville ☑ Sort
○ Valencia
○ Zaragoza

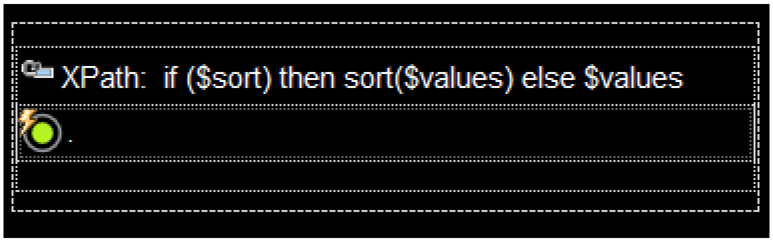
## How it works

To implement the sort functionality, we did the following:

- Added an optional **`$sort`** parameter to both control templates.
- In the design of the top page, added a check box control on the right-hand side of each placeholder control *(see screenshot below)*, and set the check box's column-width to `wrap_content`. We also set the `Visible` property of each check box appropriately.
- In the **`$PERSISTENT`** tree, added two new attributes, **`SortCountries`** and **`SortCities`**, and set the default fixed value of each to **`0`** *(see screenshot below)*. We then set these two nodes to be updated by the respective check-box value (selected or not-selected); this was done by dragging each node on to its respective check box.



- For each placeholder, defined the value of the **`$sort`** parameter to be the XPath expressions, respectively, `if ($PERSISTENT/Root/`**`@SortCountries=1`**`) then true() else false()` and `if ($PERSISTENT/Root/`**`@SortCities=1`**`) then true() else false()`.
- In each control template, the sequence that generates the repeating rows or repeating columns of the table is defined by the XPath expression `if ($sort) then sort($values) else $values`. In plain language, this means: If the value of the **`$sort`** parameter is `true()`, then sort the items in the **`$values`** sequence, otherwise use the **`$values`** sequence as it is (which is the order in the XML data file **`CitiesWorldwide.xml`**).



XPath: if ($sort) then sort($values) else $values

.

What happens is this:

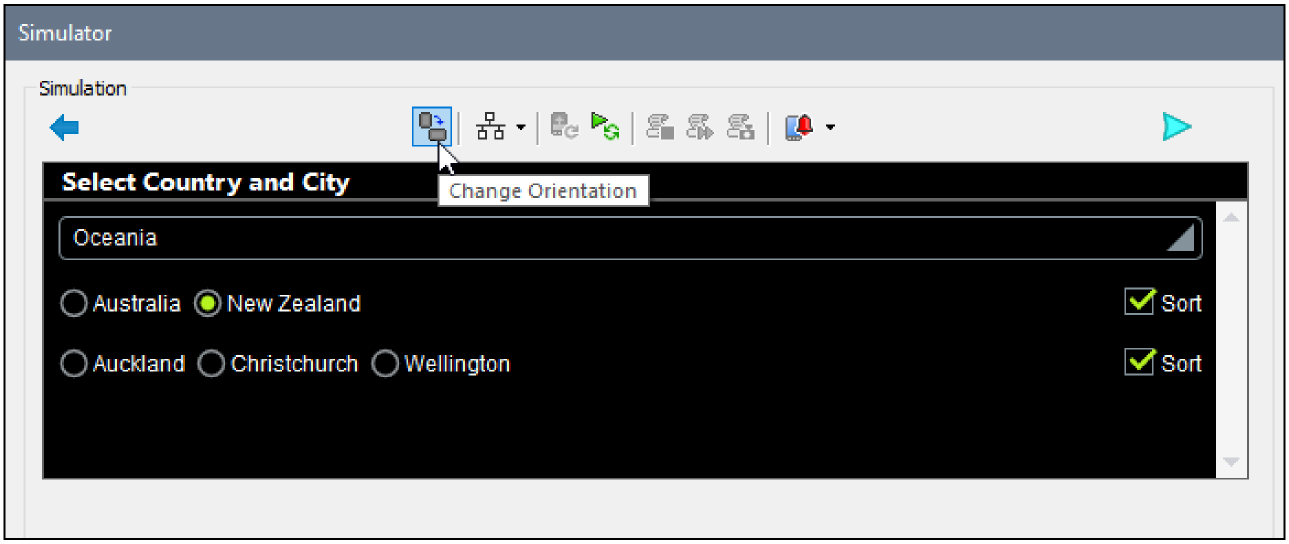
- The values of the **`SortCountries`** and **`SortCities`** attributes are set to an initial value of **`0`**. The values of both can be changed by selecting (value set to **`1`**) or deselecting (value set to **`0`**) their respective check boxes.
- The values of the respective attribute nodes (**`SortCountries`** and **`SortCities`**) are passed via the two placeholders to the **`$sort`** parameter of the respective control template as either `true()` or `false()`.
- The XPath expression that selects the items of the **`$values`** sequence—which produces the repeating rows or repeating columns—either sorts the sequence or not, depending on the value of the template's **`$sort`** parameter (`true()` or `false()`).

## 12.10.4.4 Dynamically Selecting a Template

A powerful benefit of using control templates is that you can cause a template to be selected dynamically, depending on the runtime situation. For example, a different template can be selected according to user input or according to the device's environment or settings.

In our example file `Cities4-DynamicSelection.mtd`, a horizontal or vertical template is selected according to the orientation of the mobile device (landscape or portrait). If the device is held in landscape orientation, then horizontal radio buttons are displayed **automatically**. If the user changes the device's orientation to portrait, then the display of radio buttons is **automatically** changed to a vertical display. This design feature has been added to the previous example 1219, `Cities3-DynamicUpdates.mtd`.

**Note:** You can try out this example by turning the device (between landscape and portrait orientations) while the solution is running or by simulating a change of device orientation. To simulate a change of orientation while a simulation is running, click the **Change Orientation** toolbar button *(see screenshot below)*.



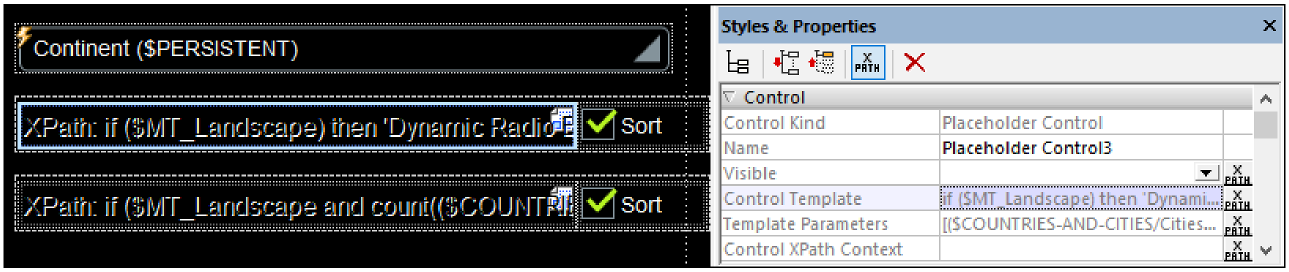
The screenshot above shows a simulation of `Cities4-DynamicSelection.mtd` in landscape orientation. In it, the control template *Dynamic Radio Buttons (Horizontal)* has been automatically selected (because of the landscape orientation) for both sets of radio buttons, and, as a result, the device automatically displays radio buttons horizontally.

**Note:** The device orientation is known to us because it is stored in the two MobileTogether variables **`$MT_Landscape`**1304 and **`$MT_Portrait`**1304 (each of which have a value of either `true()` or `false()`).

## How it works

In order to dynamically select templates, we need, in each Placeholder Control, to define two settings:

- an XPath expression to specify the conditions under which the alternative templates are respectively selected *(the relevant property is highlighted in the Styles & Properties Pane*274 *in the screenshot below)*
- an XPath expression to pass parameter values to the selected control template


Continent ($PERSISTENT)
XPath: if ($MT_Landscape) then 'Dynamic Radio
Sort
XPath: if ($MT_Landscape and count(($COUNTRI
Sort
Styles & Properties
Control

| | |
|---|---|
| Control Kind | Placeholder Control |
| Name | Placeholder Control3 |
| Visible | |
| Control Template | if ($MT_Landscape) then 'Dynami... |
| Template Parameters | [($COUNTRIES-AND-CITIES/Cities... |
| Control XPath Context | |

*Selecting the control template*

The XPath expression for the placeholder that instantiates radio buttons for countries is:

```
if ($MT_Landscape)
then 'Dynamic Radio Buttons (Horizontal)'
else 'Dynamic Radio Buttons (Vertical)'
```

This expression specifies that if the **$MT_Landscape** global variable has a value of `true()`, then the control template named *Dynamic Radio Buttons (Horizontal)* must be used, otherwise (when **$MT_Landscape**=`false()`) the control template named *Dynamic Radio Buttons (Vertical)* must be used. Remember that this expression must evaluate to the **name of the control template** to be called, which it now does for each orientation (landscape and portrait).

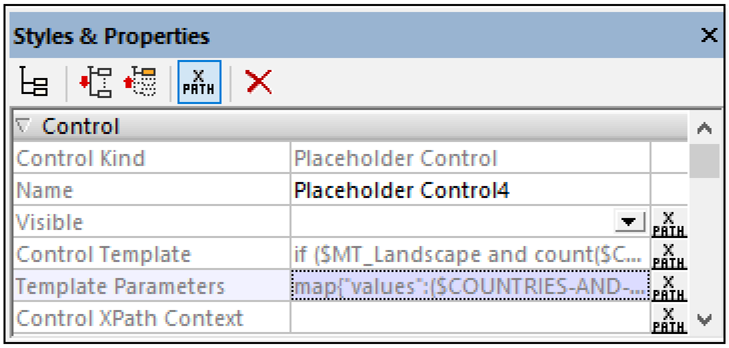
The XPath expression for the placeholder that instantiates radio buttons for cities is slightly different. Since some countries display a large number of cities (Italy 9 and USA 12), all cities of such countries might not be able to be displayed horizontally in a single screen width, even in landscape orientation. So we have restricted the selection of the horizontal template (in landscape orientation) to only those countries having less than 9 cities. The XPath expression is listed below, with the part that counts the number of cities of the selected country being highlighted in yellow.

```
if ($MT_Landscape and
count($COUNTRIES-AND-
CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent ]/Country[@name =
$PERSISTENT/Root/@Country]/City/@name) lt 9)
then 'Dynamic Radio Buttons (Horizontal)'
else 'Dynamic Radio Buttons (Vertical)'
```

The effect of the definitions listed above is that: (i) in landscape orientation, all lists of countries are horizontal, whereas in portrait orientation they are all vertical; (ii) in landscape orientation, lists of 1 to 8 cities are displayed horizontally while lists containing more than 8 cities are displayed vertically; in portrait orientation all city lists are displayed vertical.

*Defining the parameter values to pass to templates*

The XPath expression to pass values to the parameters of the selected control template *(see screenshot below)* must be either **an array expression or a map expression**. Bear in mind that you do not know which template will actually be called at runtime. However, you do know that the choice is between two templates and that they both happen to have two parameters with the exact same names. Because both parameters have the same name, the XPath expression to assign values is simpler.



Control

| | |
|---|---|
| Control Kind | Placeholder Control |
| Name | Placeholder Control4 |
| Visible | |
| Control Template | if ($MT_Landscape and count($C... |
| Template Parameters | map{"values":($COUNTRIES-AND-... |
| Control XPath Context | |

For the first placeholder, we have used an array expression *(listed below)*. The array must consist of two sequences (because we have two parameters), with the first sequence (highlighted in yellow below) providing the value of the first parameter and the second sequence (highlighted in blue) providing the value of the second parameter. The first sequence generates the value of the **$values** parameter and will evaluate to a list of countries of the selected continent. The second sequence generates the value of the **$sort** parameter and will evaluate to `true()` or `false()`. Note that the order of the sequences in the array must correspond to the order of the parameter definitions in the project, and must include values for any optional parameters—which can be the empty sequence `()` if no value is needed.

```
[
($COUNTRIES-AND-CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent]/Country/@name),
(if ($PERSISTENT/Root/@SortCountries=1) then true() else false())
]
```

For the second placeholder, we have used a map expression *(listed below)*. The map must consist of **`key:value`** pairs, where the key must be the name of the parameter. The **`key:value`** pairs can be supplied in any order.

```
map{
"values":($COUNTRIES-AND-
CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent ]/Country[@name =
$PERSISTENT/Root/@Country]/City/@name),
"sort":if ($PERSISTENT/Root/@SortCities=1) then true() else false()
}
```

## 12.11 Rich Text

The Rich Text feature enables an XML page source that contains rich text formatting to be displayed in a solution so that the formatting is retained. You can also define your own styling for different elements. On Web clients and Windows clients, such content can also be edited and saved back to the XML page source so that new content also has rich text formatting and is saved with the appropriate XML mark up.

The feature is implemented using the following mechanism:

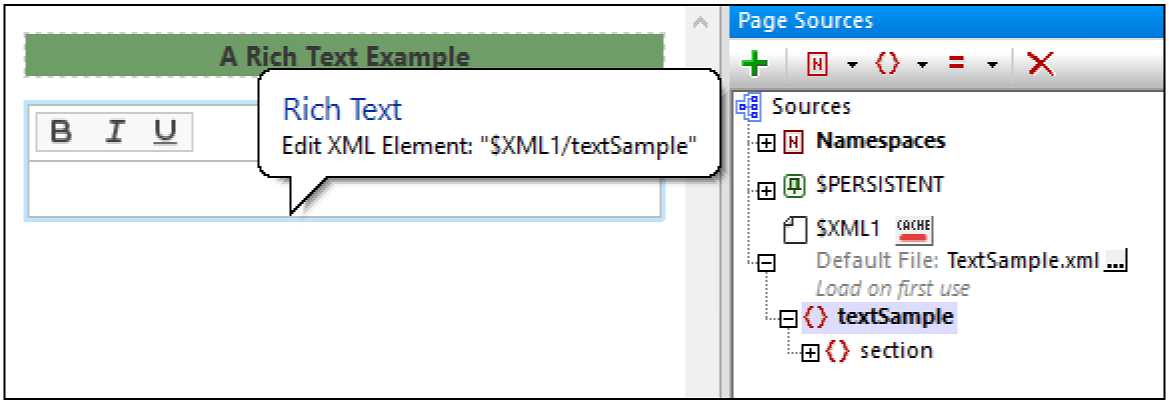
- A Rich Text control 1225 is placed at the location in the design where you want to display the rich text. The control has two key associations: (i) with the XML page source; (ii) with a Rich Text style sheet 1226 .
- In the design's Rich Text Style Sheets dialog 1228, you can define style rules for different elements and style mappings for existing style rules.

*In this section*
This section is organized into the following sub-sections:

### 12.11.1 The Rich Text Control

The first step towards enabling Rich Text in your design is to add the Rich Text control 584 at the location in your page where you want to display the text. Set up the control as follows:

1. Drag and drop the Rich Text control 584 on to your page (*see screenshot below*).
2. Drag and drop the **root element** of the page source containing your rich text onto the control. (Note that the control's page source link must be the root element of the page source; the control cannot be linked to an attribute or to any other element than the root element.) For example, in the screenshot below, the control has been linked to the root element of **`$XML1`**, which is the element `textSample`. (The rich text that we want to display is stored in this XML page source.)



3. Open the Rich Text Style Sheets dialog 1598 via the **Project | Rich Text Style Sheets** menu command, and create a new style sheet 1226. You can define multiple style sheets for the project. This enables you to assign different style sheets to different Rich Text controls 584. How to set up a style sheet is described in the section Rich Text Style Sheets 1226.
4. After you have created the style sheet, select the Rich Text control 584 in the design and, via its `Rich Text Style Sheet` property, assign the style sheet to the control. Note that each Rich Text control 584 can have only one style sheet assigned to it at a given time.

**Note:** The height of the control can be set with the control's **`Rich Text Height`** 584 property.

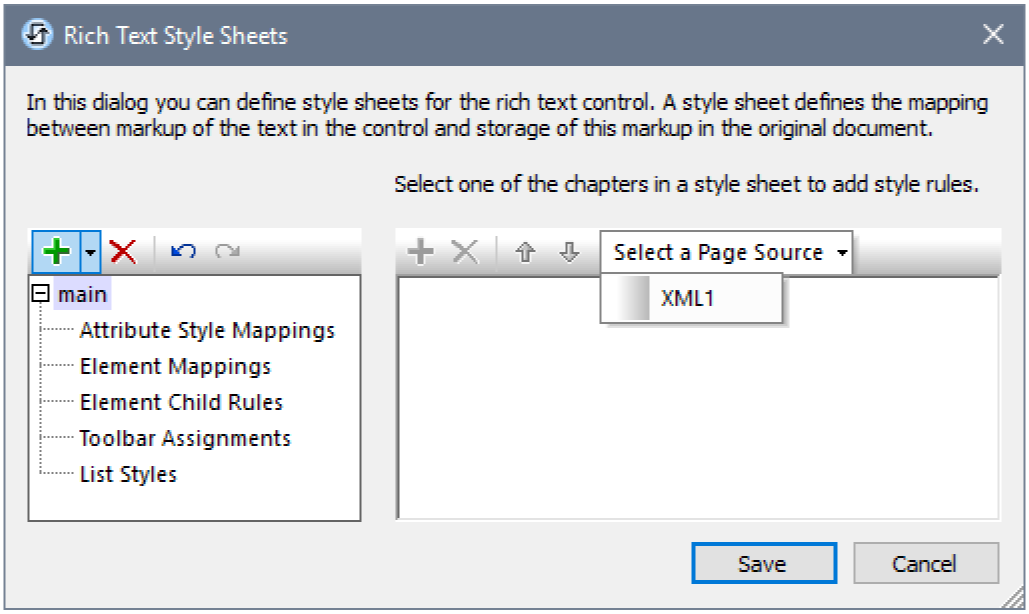
## 12.11.2 Rich Text Style Sheets: Setup

A Rich Text style sheet consists of multiple rules that describe the transformation, in both directions, between XML (data in the page source) and HTML (for the display in the Rich Text control 584). You can define multiple style sheets for a project. Any one of these style sheets can be assigned to a Rich Text control 584. The styles in that style sheet are then used for the styling of the text displayed in that Rich Text control 584.

### Create a style sheet

To create a Rich Text style sheet, do the following:

1. Select the menu command **Project | Rich Text Style Sheets** 1598. The Rich Text Style Sheets dialog (*see screenshot below*) appears.
2. Click **Add Style Sheet** in the toolbar of the left pane to add an empty style sheet. Alternatively, you can add a style sheet with predefined styles for HTML elements by clicking the dropdown arrow of **Add Style Sheet**, and selecting **Add HTML Style Sheet**. See the next section below 1227 for a description of this style sheet.
3. Edit the name of the style sheet to something suitable for your project. In the screenshot below, for example, the style sheet has been renamed **`main`**.
4. Optionally, select a page source that has the element structure you want to use. (The elements of this page source will be used to provide the entry helper items that are displayed during editing.)
5. Click **Save** to save the style sheet with the project.



In this dialog you can define style sheets for the rich text control. A style sheet defines the mapping between markup of the text in the control and storage of this markup in the original document.

Select one of the chapters in a style sheet to add style rules.

How to create style mappings and rules is described in the section Rich Text Style Sheets: Styles[1228].

## Predefined HTML style sheets

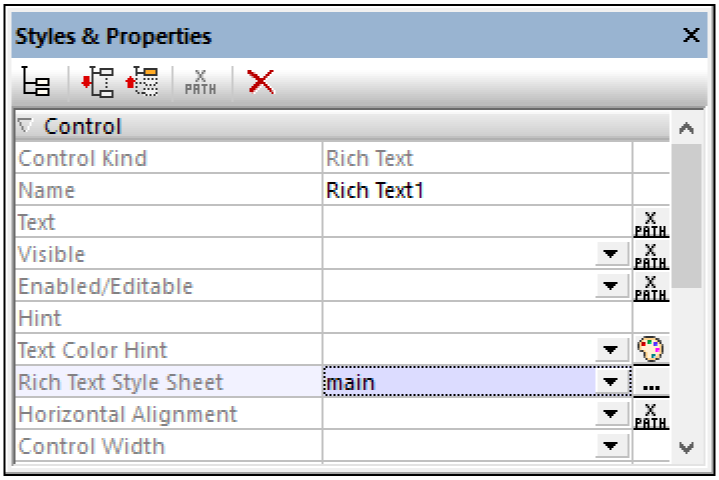
When adding a Rich Text style sheet, you can choose to add an HTML style sheet (*see Point 2 of the procedure listed above*). This style sheet contains predefined styles for the following elements (and also toolbar assignments):

- In the Element Mappings[1229] section, styles for the following commonly used HTML inline elements are defined: **b**, **strong**, **i**, **em**, **u**, **ins**, **del**, **mark**, **small**, **sub**, **sup**.
- In the List Styles[1232] section, styles for list-related HTML elements are defined.
- In the Toolbar Assignments[1231] section, the HTML elements **b**, **i**, **u**, **del** have been assigned to toolbar icons.

You can modify the existing styles and/or add new styles as required. How to do this is described in the section Rich Text Style Sheets: Styles[1228].

## Assign the style sheet to the Rich Text control

After the style sheet has been created and saved with the project, you can assign it to the Rich Text control[584] via the control's `Rich Text Style Sheet`[584] property. In the screenshot below, for example, the Rich Text control[584] has been assigned the style sheet named **main**.



Styles & Properties

Control

| | |
|---|---|
| Control Kind | Rich Text |
| Name | Rich Text1 |
| Text | |
| Visible | |
| Enabled/Editable | |
| Hint | |
| Text Color Hint | |
| Rich Text Style Sheet | main |
| Horizontal Alignment | |
| Control Width | |

## 12.11.3 Rich Text Style Sheets: Styles

After a Rich Text style sheet has been created 1226, style rules and mappings are defined in different sections of the Rich Text Style Sheets dialog. These rules and mappings are concerned firstly with the conversion of styles in the XML of the page source into HTML that can be displayed on client devices. In the case of web clients and Windows clients, rich text content can be edited and formatted by the end user. The rules and mappings are therefore also used to pass any modified HTML styles back to the XML page source.

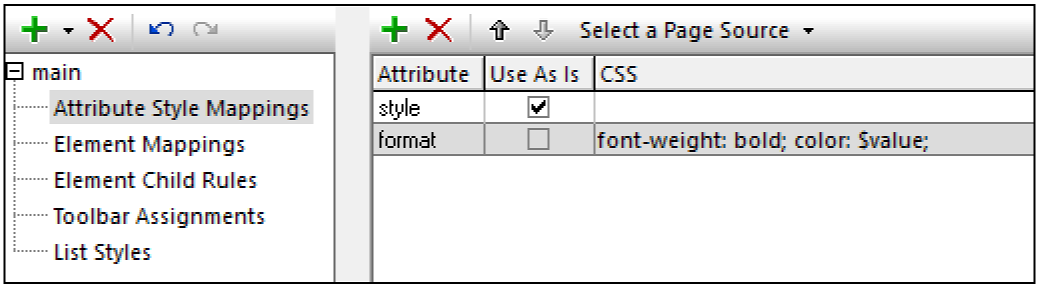
In the Rich Text Style Sheets dialog, the style rules and mappings are organized into sections. Each of these sections is described below:

- Attribute Style Mappings 1228
- Element Mappings 1229
- Element Child Rules 1230
- Toolbar Assignments 1231
- List Styles 1232

### Attribute Style Mappings

The attributes defined in this section (*see screenshot below*) map attributes of the same name in the page source to content in the Rich Text control 584. The mappings apply to attributes of **all** elements in the page source. Any attribute defined here describes how its parent element will be styled. Conversely, any attribute that is **not defined here** does **not pass any styling** to its parent element.

You can add an attribute to the list of styled attributes by clicking **Add Style** in the toolbar of the right-hand pane and entering the name of the attribute.



- main
  - Attribute Style Mappings
  - Element Mappings
  - Element Child Rules
  - Toolbar Assignments
  - List Styles

| Attribute | Use As Is | CSS |
|---|---|---|
| style | ☑ | |
| format | ☐ | font-weight: bold; color: $value; |

Note the following points:

- *Use As Is* expects content of the attribute in the page source to be valid CSS.
- *CSS:* Takes one or more CSS property–value pairs. If you want to use a property value from an attribute in the XML page source, use `$value` to get the value of that attribute. For example, if an element in the page source has an attribute named `format` so that: (i) `<myelement format="red">...</myelement>` and (ii) the `format` attribute is defined in the style sheet as shown in the screenshot above, then the entire value of the `format` attribute, that is `red`, replaces `$value` in the style definition. So, for `myelement`, the style definition (obtained from the style definition of the `format` attribute) would resolve to: `font-weight:bold; color:red`. If another element had `@format="blue"` in the page source, then that element's style definition would resolve to: `font-weight:bold; color:blue`.
- If an element in the XML page source has two attributes for both of which *Use As Is* style definitions exist in the style sheet, then the two style definitions are combined for styling that element.

*Examples*

- If the attribute definitions shown in the screenshot above are applied to the following element in the XML of the page source:
  `<heading style="font-style:italic;" global="font-weight:bold;">Text Formatting</heading>`
  then the resulting style definition will be **`font-style:italic;`** since the **`@style`** attribute is used as is. The **`@global`** attribute is ignored because it is not defined in the style sheet.

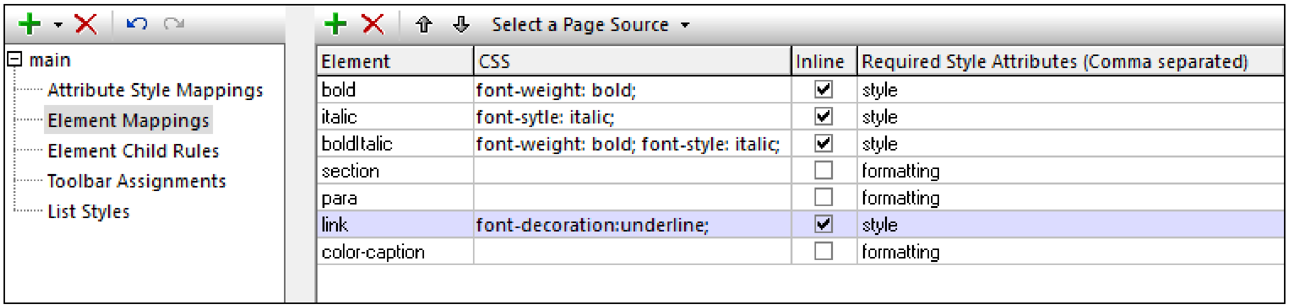
- If the attribute definitions shown in the screenshot above are applied to the following element:
  `<heading format="red">Text Formatting</heading>`
  then the style that will be applied will be **`font-weight:bold; color:red;`** because this is the definition of the **`format`** attribute in the style sheet.

## Element Mappings

Each element mapping (*see screenshot below*) defines certain properties for the listed elements. You can do the following:

- Define the CSS style properties of the element. Note that the style properties of elements can also be defined via attributes in the *Attribute Style Mappings* 1228 section; avoid defining the same property in two locations.
- Specify whether an element is a block element (corresponds to HTML **`div`**) or an inline element (corresponds to HTML **span**). The default setting is block; so, by default, the content of every element would appear in the solution on a new line unless specified here as being inline. Inline elements, on the other hand, occur within a line. Common inline elements are those that mark up text within a line as bold or italic. In the screenshot below, note which elements have been specified as being inline.

- Specify what attribute/s of an element will hold the element's style properties. These attributes will be the attributes in the XML document to which styling properties are written (from the client).
- If a block element has been defined to have an attribute to hold styles (*see previous point*), then, when the end user places his or her cursor inside this element's content, the text-alignment icons (left, center, right, justify) are automatically enabled in the control's toolbar 1233. If the end user applies text-alignment to the element's content by tapping a text-alignment icon, then that icon's style value is written to the attribute that was defined to hold the element's style properties. For example, if the end user places the cursor in the text of the **`color-caption`** element and taps the **Right-align** icon, then the text will be right-aligned in the control and a CSS property-value of **`text-align:right`** will be stored to the **`formatting`** attribute of that **`color-caption`** element—if, as in the screenshot below, it is **`formatting`** that is specified as the attribute to hold the **`color-caption`** element's style properties.



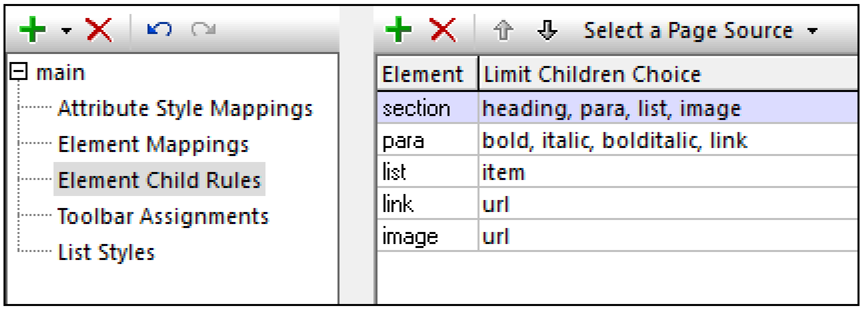
| Element | CSS | Inline | Required Style Attributes (Comma separated) |
|---|---|---|---|
| bold | font-weight: bold; | ☑ | style |
| italic | font-sytle: italic; | ☑ | style |
| boldItalic | font-weight: bold; font-style: italic; | ☑ | style |
| section | | □ | formatting |
| para | | □ | formatting |
| link | font-decoration:underline; | ☑ | style |
| color-caption | | □ | formatting |

You can add an element to the list by clicking **Add Style** in the toolbar of the right-hand pane and entering the name of the element you want to define.

## Element Child Rules

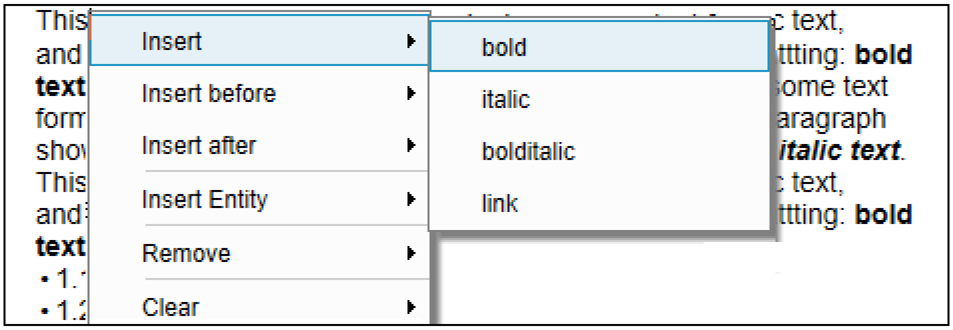
The rules in this section (*see screenshot below*) define what child elements an element can have. The list of child elements you enter does not need to match the complete list of all child elements allowed by the schema; it is the list of child elements that you wish to allow the end user to add when editing content in the Rich Text control 584. If an element is not listed here, then all elements will be available as children of that element.

You can add an element to the list by clicking **Add Style** in the toolbar of the right-hand pane and entering the name of the element you want to define. In the next column, enter a comma-separated list of the names of child elements.



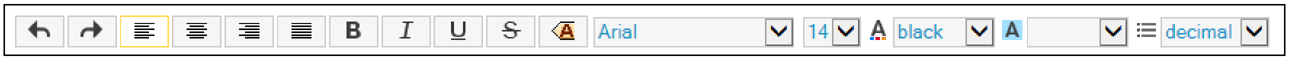
| Element | Limit Children Choice |
|---|---|
| section | heading, para, list, image |
| para | bold, italic, bolditalic, link |
| list | item |
| link | url |
| image | url |

In the screenshot above, for example, a **`para`** element is defined to have the following children: **`bold`**, **`italic`**, **`bolditalic`**, and **`link`**. If the end user, while editing content in a Rich Text control that uses this style sheet, right-clicks inside the **`para`** element, then the context menu shown in the screenshot below appears. The elements that can be inserted are those that were defined as children of the **`para`** element.



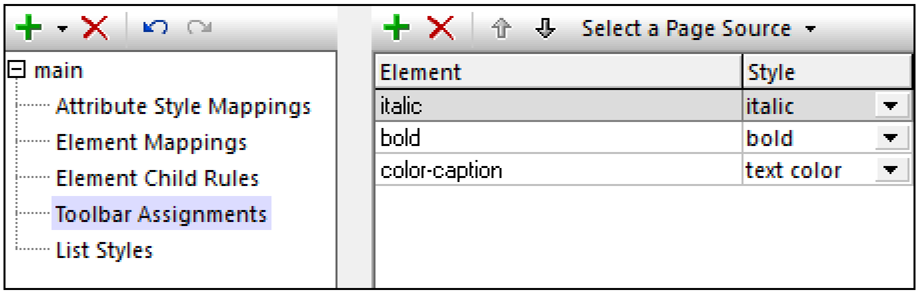
## Toolbar Assignments

When the Rich Text control is displayed in the solution of a web client or Windows client, the content that is displayed in the control is editable. On these devices, an editing toolbar (*see screenshot below*) is displayed in the top part of the control. The end user can click an icon in the toolbar to assign that icon's style selection to the selected text content.



It is in the *Toolbar Assignments* section of the Rich Text Style Sheets dialog (*screenshot below*) that you map XML elements to toolbar items. In the screenshot below, for example, (i) the **italic** icon (selected in the combo box in the *Style* column) is mapped to an element named **`italic`**; (ii) the **bold** icon is mapped to an element named **`bold`**; and (iii) the **text color** selection is mapped to an element named **`color-caption`**. When the end user clicks a toolbar icon or makes a selection in one of the combo boxes of the toolbar, then the styling associated with that toolbar item is applied to the selected text.

At the XML level, the element that has been mapped to that toolbar item is created around the selected text; if this element does not have any style associated with it, then an attribute containing the relevant style is added to the element. For example, in the screenshot below, since the **`italic`** element has been assigned an italic style in the Element Mappings 1229 section of the dialog, creating the **`italic`** element around a text selection (by clicking the **italic** toolbar icon) will cause the text selection to be displayed in italic. In the case of the **text color** toolbar item shown in the screenshot below, no style property has been defined for the **`color-caption`** element (*see the screenshot in the Element Mappings 1229 section*) . As a result, the text color that the end user selects in the toolbar will be stored in the **`formatting`** attribute of the **`color-caption`** element. To which attribute of an XML element styles from the edited solution are mapped is specified in the element's definition in the Elements Mapping 1229 section (see the definition of **`color-caption`** in the screenshot there).



| Element | Style |
|---|---|
| italic | italic |
| bold | bold |
| color-caption | text color |

You can add an element to the *Toolbar Assignments* list by clicking **Add Style** in the toolbar of the right-hand pane and entering the name of the element. In the next column, select the corresponding toolbar item from the combo box. (All the available toolbar items are listed in the dropdown list of the combo box.)

**Note:** When the cursor is placed at a location where a toolbar-based style cannot be applied, then that toolbar item is disabled.

**Note:** When a deployed solution is opened on a client for editing, toolbar items that have not been mapped to an element[1231] are **not displayed**. In the designer (during simulations[1355]), however, these toolbar items **are displayed**, but they are disabled and shown with a red border; if you place the cursor over one of these toolbar items, a tool tip informs you that a mapping for that toolbar item has not been defined. In this way, you can distinguish between toolbar items that are disabled because of the cursor location (*see previous note*) and toolbar items that are disabled because they have not been mapped.

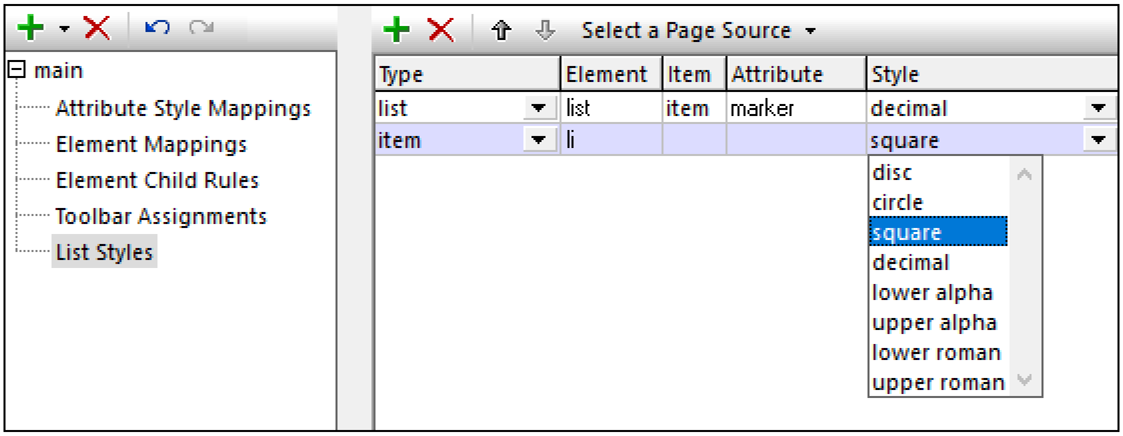
## List Styles

List styling is available for two types of list structure:

- *Container list (editable marker applies to entire list):* A list structured as a list element containing list-item child elements. For example: **`<list> <item/>...<item/> </list>`**. In the screenshot below, the first definition is that of a list of this type. The end user can change the list marker character (for example, whether the marker is a number or a square) of the entire list. You, as the designer, can define the initial marker of the list.
- *Item-sequence list (editable marker applies to individual list items separately):* A list consisting of a sequence of list-item elements, with no containing list element. For example: `<li>`**aaa**`</li>...<li>`**nnn**`</li>`. In the screenshot below, the second definition is that of a list of this type. The marker characters of **individual** list items can be edited by the end user; **all** the markers **cannot** be edited together at once.

The end user can edit the **content** of both types of list, and add and delete list items (*see Editing Rich Text Content*[1233]).

**Note:** The selected marker determines whether the list is a numbered list or itemized list.



| Type | Element | Item | Attribute | Style |
|---|---|---|---|---|
| list | list | item | marker | decimal |
| item | li | | | square |

To define list styles (*refer to the screenshot above*), do the following:

1. Add an entry for a list by clicking **Add Style** in the toolbar of the right-hand pane.

2. In the *Element* column: for Container lists, enter the name of the element that corresponds to the list; for Item-sequence lists, enter the name of the list-item element.
3. In the *Item* column: for Container lists, enter the name of the element that corresponds to the list item; for Item-sequence lists, enter nothing.
4. In the *Attribute* column: for Container lists, enter the name of the attribute that will hold the name of the list marker; for Item-sequence lists, enter nothing. When the end user selects some other marker for a Container list than the initial marker, the new marker name is entered in this attribute, and the list is styled on the basis of this attribute's value.

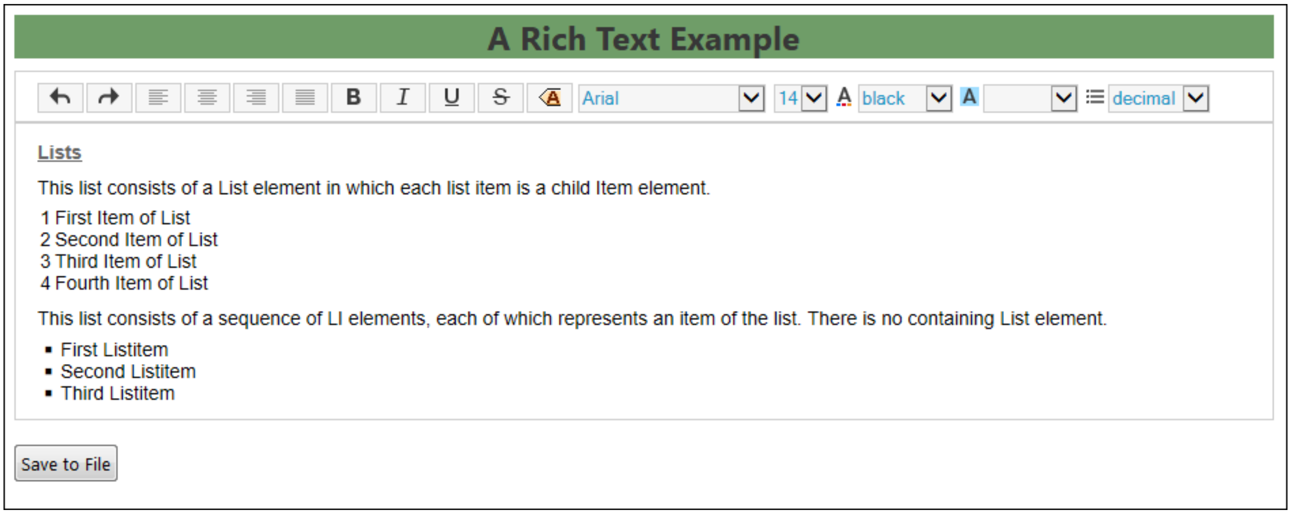
## 12.11.4 Editing Rich Text Content

Text content in a Rich Text control 584 is **displayed** on all client devices. The styles are defined in the style sheet that is assigned to the control. On Web clients and Windows clients, the end user can additionally edit the text content. This section describes how to edit rich text on a Web or Windows client.

### The Rich Text control

When a Rich Text control is rendered on a Web or Windows client (*screenshot below*), it will consist of two parts:

- A toolbar for editing, which is fixed at the top of the control.
- A content part, in which text can be edited (modified, added, or deleted). If the content requires more space than is available for the control on the device, then the content part of the control will have a scroll bar. The height of the control can be set with the control's `Rich Text Height` 584 property.



Lists

This list consists of a List element in which each list item is a child Item element.

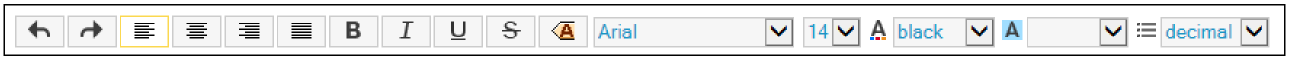
1 First Item of List
2 Second Item of List
3 Third Item of List
4 Fourth Item of List

This list consists of a sequence of LI elements, each of which represents an item of the list. There is no containing List element.

- First Listitem
- Second Listitem
- Third Listitem

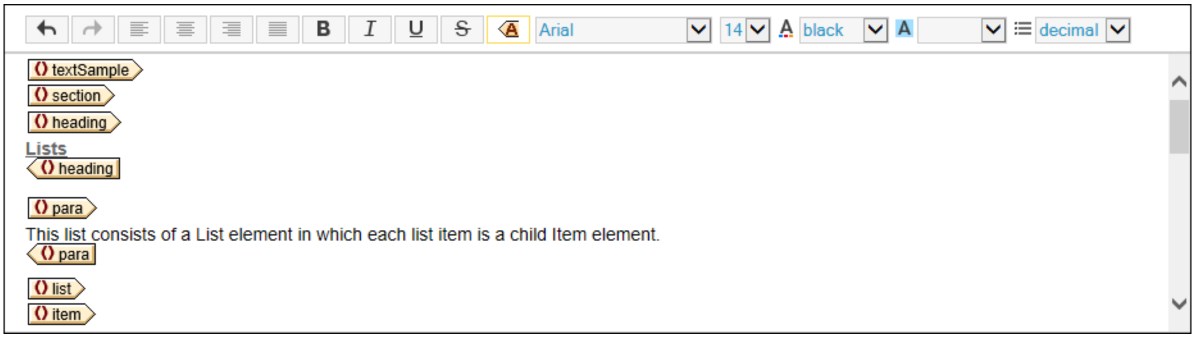
### The control's toolbar

The toolbar of the Rich Text control 584 (*screenshot below*) is displayed on Web and Windows clients only.



It consists of the following icons (from left):

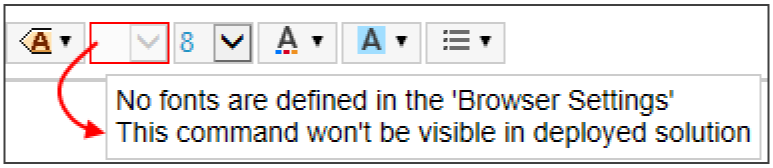
- An *Undo* (**Ctrl+Z**) icon and a *Redo* (**Ctrl+Y**) icon.
- Text-alignment icons: respectively, to align text left, center, and right, and to justify text. These icons will become enabled when the cursor is placed inside a block element for which a style attribute has been defined[1229].
- Icons to apply bold (**Ctrl+B**), italic (**Ctrl+I**), underline (**Ctrl+U**), and strike-through formatting to the selected text. At the markup level, the respective styles are applied by enclosing the selected text with the element that is assigned to that icon. (Either the element or a designated attribute will specify the respective style, see Toolbar Assignments[1231].)
- You can cut (**Ctrl+X**), copy (**Ctrl+C**), and paste (**Ctrl+V**) content. If content was copied from a Rich Text control, a pop-up will appear when pasting, asking whether you want to paste the content as XML or as text. The copied XML tree will be pasted as XML if the XML structure at that point allows the copied elements to be inserted; otherwise only the text content is pasted.
- An icon with a dropdown menu of markup tag sizes. You can select sizes from between a maximum, which shows the entire node name, to a minimum, which shows empty small start and end tags. The screenshot below shows markup tags at maximum size.



Lists

This list consists of a List element in which each list item is a child Item element.

- Four combo boxes in which to select, respectively, the font family, font size, font color, and font background. The set of available fonts can be customized in the Browser Settings[296] dialog. If no font is specified in the Browser Settings[296] dialog, then the font selection combo box will be disabled in the deployed solution. A warning to this effect is displayed in simulations when the cursor is placed over the font selection combo box (*see screenshot below*).



No fonts are defined in the 'Browser Settings'
This command won't be visible in deployed solution

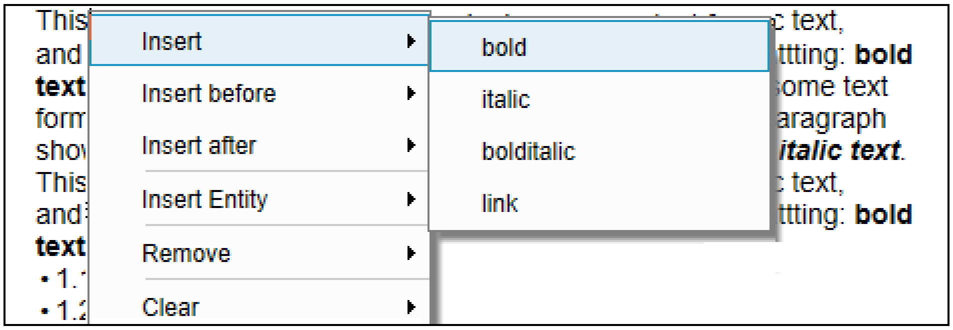
- A combo box in which to select the list-item marker of (i) the entire list (in the case of Container lists[1232]), or (ii) the selected list-item (in the case of Item-sequence lists[1232]).

**Note:** When the cursor is placed at a location where a toolbar-based style cannot be applied, then that toolbar item is disabled.

**Note:** When a deployed solution is opened on a client for editing, toolbar items that have not been mapped to an element[1231] are **not displayed**. In the designer (during simulations[1355]), however, these toolbar items **are displayed**, but they are disabled and shown with a red border; if you place the cursor over one of these toolbar items, a tool tip informs you that a mapping for that toolbar item has not been defined. In this way, you can distinguish between toolbar items that are disabled because of the cursor location (*see previous note*) and toolbar items that are disabled because they have not been mapped.

## Inserting and removing elements

To insert an element in the text content, it is best to define child elements of editable elements[1230] in the style sheet. If this has been done, then the end user can right-click a text selection inside an element and, from the context menu that appears, select the element to insert (*see screenshot below*).

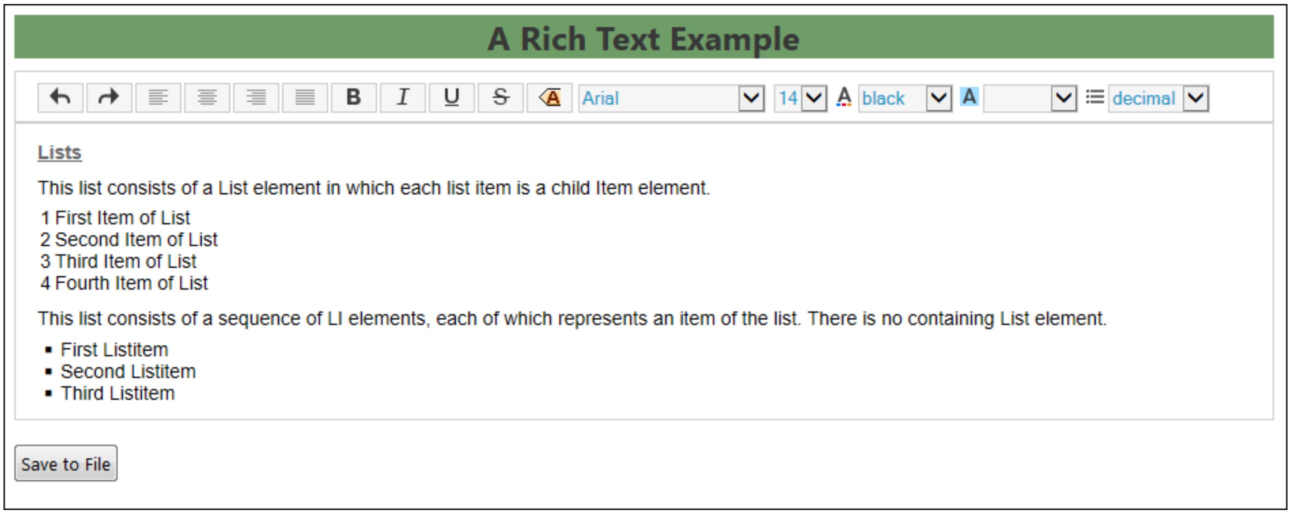


The elements that are shown in the context menu depend on the element in which the cursor is placed:

- *Insert:* Inserts a child element at the cursor selection point or around a text selection. The context menu lists the child elements that have been defined.
- *Insert before/after:* Inserts a sibling element before or after the element in which the cursor is placed. The context menu shows the sibling elements based on the structure of the page source selected in the style sheet.
- *Insert Entity:* Inserts XML special characters (ampersand, apostrophe, less-than character, greater-than character, quote) as character entities at the cursor selection point or as a replacement of the text selection.
- *Remove:* Removes the element in which the cursor is placed or an ancestor element. The context menu shows the current element and all ancestors based on the structure of the page source selected in the style sheet.
- *Clear:* Clears markup that has been added to enclose a text selection. For example, if a text selection has been made bold, then the markup that created the bold styling is cleared.

## Editing a list

The following screenshot shows two lists. The first list is a Container list[1232] while the second list is an Item-sequence list[1232].



Lists

This list consists of a List element in which each list item is a child Item element.

1 First Item of List
2 Second Item of List
3 Third Item of List
4 Fourth Item of List

This list consists of a sequence of LI elements, each of which represents an item of the list. There is no containing List element.

- First Listitem
- Second Listitem
- Third Listitem

Note the following list-specific editing functionality:

- Change the list-item marker via the List Marker combo box in the toolbar. If an item in a Container list 1232 is selected, then the markers of all items in the list are changed. If an item in an Item-sequence list 1232 is selected, then the marker of only that list item is changed.
- To insert a list item, place the cursor at the end of the list item immediately below which you want to insert the new list item, and press **Enter**.
- To remove a list item, toggle on the markup tags and select the list item you want to remove; then select **Remove** in the context menu.

# 12.12 Solutions for Authenticated Users

Consider the following situation. An end user is authenticated (correctly logged in) on one MobileTogether Server and is running a solution from that server. You, as the designer, wants to start a solution from another MobileTogether Server on that user's device. Since the user is already authenticated on one MobileTogether Server, you would like the solution on the second MobileTogether Server to be started directly, without the user having to log in to the second MobileTogether Server. MobileTogether enables you to securely pass authentication information to a solution on another MobileTogether Server.

**Note:** The authentication described in this topic applies only to solutions running on web clients.

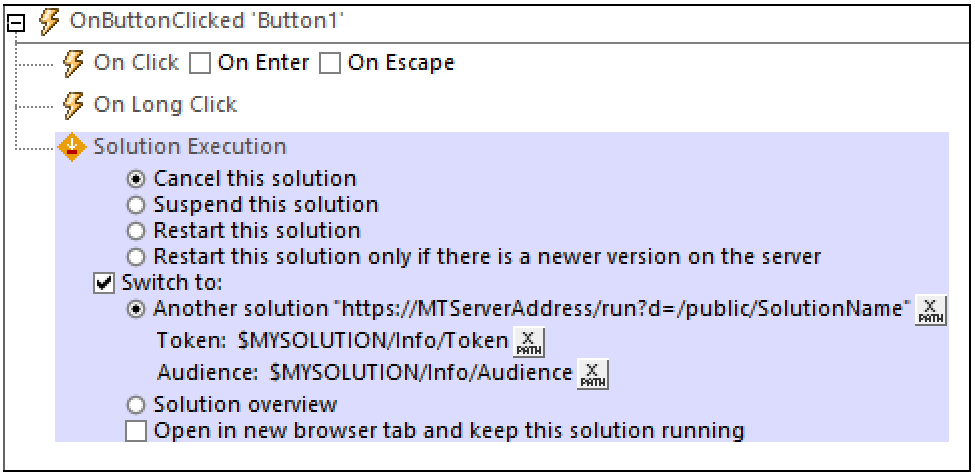
## Setting up authenticated users for remote solutions

The steps for setting up the transfer of user authentication to a solution on another MobileTogether Server are given below.

### Terminology note

- *Authentication Host:* The first MobileTogether Server, on which the authentication has already been carried out.
- *Authentication Receiver:* The second MobileTogether Server, from which the second solution is served and on which we want authentication for the solution to be automatic.

1. In the MobileTogether Server settings of the Authentication Receiver, enter and enable the settings of the Authentication tab. These settings are: (i) the address of the Authentication Host, (ii) the secure HTTPS port for mobile clients, and (iii) the *Audience* string, which is a unique string that identifies the audience of this MobileTogether Server. **Note also:** (i) that both servers must use SSL encryption (HTTPS connections), and (ii) that both solutions (the calling and called) must be run for anonymous users.
2. In the calling solution, define a Solution Execution action 915 *(see screenshot below)* at an appropriate point during solution execution.


OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Solution Execution
● Cancel this solution
○ Suspend this solution
○ Restart this solution
○ Restart this solution only if there is a newer version on the server
☑ Switch to:
● Another solution "https://MTServerAddress/run?d=/public/SolutionName"
Token: $MYSOLUTION/Info/Token
Audience: $MYSOLUTION/Info/Audience
○ Solution overview
□ Open in new browser tab and keep this solution running

The relevant settings of the action are:

- *The solution's address:* Which should evaluate to an address of the form `https://MTServerAddress/run?d=/public/SolutionName`.
- *Token:* A user-defined XML tree that is securely passed to the solution on the second server. You define this XML tree so that it contains all the information you want to pass. MobileTogether requires only that this entry is a well-formed XML tree.
- *Audience:* This string must match the *Audience* setting of the Authentication Receiver *(see Point 1 above)*.

3. If the audience matches, then the XML tree that is passed to the Authentication Receiver (via the *Token* setting of the Execution Solution action; *see previous point*) will be passed to the second solution in the `$MT_AuthenticationToken` variable. You can access nodes in the token via XPath expressions that use the variable (for example: **`$MT_AuthenticationToken/Root/User/@id`**). If the audience does not match, then the solution will start, but without a valid authentication token. It is left to you as the developer to decide how to handle this situation. One way would be to show a suitable message and forward the user back to the calling solution.

# 12.13 Hyperlinking to Solutions

You can create hyperlinks to solutions in the following ways:

- Via the Open URL 681 action of page or control events
- In an email that the end-user sends 693

If the URL of the hyperlink does not contain a query string, then the solution is opened at its start page. If the URL does contain a query string, then the solution is opened in accordance with the logic of the solution and the query string. As examples of the two types of URLs (without and with a query string), think about the URL of a search engine such as Google.

- This URL, without a query string, opens the Google start page: https://www.google.com/
- This URL contains a query string that queries the Google search engine for "Altova MobileTogether" (everything after the question mark is the query string). The URL directly opens a page containing the results of the search (and not the start page of Google): https://www.google.com/search?q=Altova+MobileTogether&ie=utf-8&oe=utf-8&gws_rd=cr&ei=3YAaVdDDA4SYsgGOm4A4

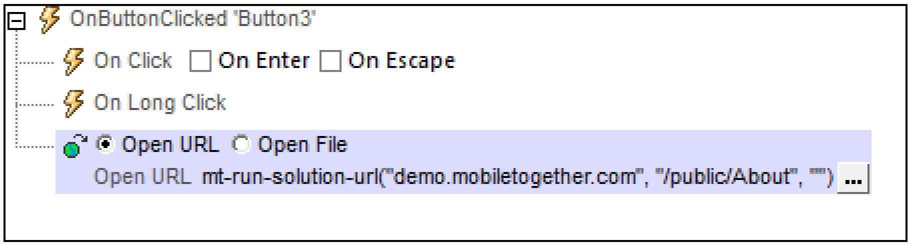
**Note:** Links to update server settings do not work in Gmail and some other email applications, but they work in popular clients such as AquaMail, K9, and MailWise. They have been tested in AquaMail and K9 and work correctly in these applications.

## Linking to a solution from a design component

A design component can be linked to a solution via the component's Open URL 681 action. For example, if a button is clicked, the button's Open URL 681 action can specify that a solution is opened.

Create a solution link as follows:

1. For the event on which you wish to specify the solution link, create an Open URL 681 action (*see screenshot below*).
2. Create an XPath expression that uses the `mt-run-solution-url` 1262 function to generate the URL of the solution. The function is described below.



Open URL mt-run-solution-url("demo.mobiletogether.com", "/public/About", "")

▼ mt-run-solution-url

**`mt-run-solution-url`**`(`**`ServerAddress?`**` as xs:string, `**`SolutionName?`**` as xs:string, `**`InputParameters?`**` as xs:string) `**`as xs:string?`**

Generates a URL to open the specified solution in a MobileTogether client. When the URL is tapped, the Altova MobileTogether Client app is opened and the solution is started in the app. The URL is generated from either (i) the function's three submitted arguments *(listed below)*, or (ii) the function's `InputParameters` argument.

- `ServerAddress`: Takes the name or IP address of the MobileTogether Server on which the solution that you want to run is deployed. If this argument is omitted or it is the empty string, then the current server is used.
- `SolutionName`: Takes the deployed path of the solution on the server. For example: `/public/MySolution` (which would point to the `MySolution.mtd` file in the `/Public` container). If this argument is omitted or it is the empty string, then the current solution is used.
- `InputParameters`: Takes the function `mt-run-solution-url-parameters` as its input. The argument of the **`mt-run-solution-url-parameters`** function is either (i) a sequence of string values that will be the values of the query's parameters, or (ii) a map of *`key:value`* pairs that provide the name and value of the respective parameters. This function returns a string that is correctly encoded and percent-escaped according to the rules for encoding URL query strings. See the description of the **`mt-run-solution-url-parameters`** function below. (Additionally, the `InputParameters` argument can be provided as a string that is already encoded for the query string part of a URL (*see fourth example below*).)

The `mt-run-solution-url` function therefore creates a URL, with or without query parameters, that accesses a solution on a MobileTogether Server. The query parameters are passed to the solution when the solution is opened via the URL. The values of these parameters can be accessed in other design components by using the **`$MT_InputParameters`**[1300] global variable.

*Examples*

- **`mt-run-solution-url`**`('100.00.000.1', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the server with the IP address `100.00.000.1`. The URL has no query parameters.
- **`mt-run-solution-url`**`('', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the current server. The URL has no query parameters.
- **`mt-run-solution-url`**`('', '', mt-run-solution-url-parameters(('2015', 'USA', 'true')))` returns a URL that points to the current solution on the current server. The argument of the **`mt-run-solution-url-parameters`** function in this example is a sequence of string values that will be the values of the query's parameters. The first string will be the value of the first parameter, the second string will be the value of the second parameter, and so on. See the description of the **`mt-run-solution-url-parameters`** function below.
- **`mt-run-solution-url`**`('', '', 'in1=value1&in2=value2%3FAndMoreValue2')` returns a URL that points to the current solution on the current server. The `InputParameters` argument is submitted as a string already encoded as a URL query string.

Note the following points:

- The first argument, `ServerAddress`, is used to look up information on the client about a server having the submitted name/address. The port number, user name, and user password that are associated with the server name are then used to connect to the server. So if a URL is generated with a server name that is not recognized by the client, then the URL will not work.
- The second argument, `SolutionName`: (i) generates the deployed path (on the server) if the solution is run on the server, but (ii) generates a file path for simulations.
- The third argument, `InputParameters`, uses the MobileTogether-specific XPath extension function called **`mt-run-solution-url-parameters`** to generate and encode the query's parameter-value pairs. Do not confuse the **`mt-run-solution-url-parameters`** function (which encodes the query parameters) with the `mt-run-solution-url` function (which generates the whole URL).

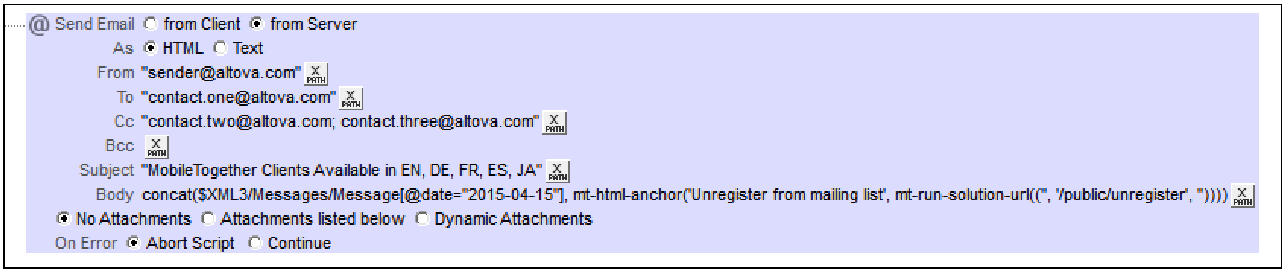
## Using hyperlink query parameter values in other design components

When a solution is opened by triggering a hyperlink that is associated with a control event or page event, any parameter values in the hyperlink's URL are passed to the solution and can then be used in other design components in the **target** solution. The values are stored by default as a map in the **`$MT_InputParameters`** global variable of the target solution.

Alternatively, you can can change the data structure of the **`$MT_InputParameters`** variable in individual projects (in the More Project Settings dialog[296]) to be a sequence of string values. If string values are passed to **`$MT_InputParameters`**, these are alphabetically sorted on the key of the URL's parameters. The order of the string values in the **`$MT_InputParameters`** sequence is the same as that in the sequence submitted to the **`mt-run-solution-url-parameters`**[1262] function for generating the URL's query parameters. Since the order of string values in the **`$MT_InputParameters`** is known to you (alphabetically sorted on the parameter keys), each string can be accessed in XPath expressions by using position predicates. For example: **`$MT_InputParameters[1]`** returns the first string value in the sequence, and **`$MT_InputParameters[2]`** returns the second string value.

## Linking to a solution from an email that the end-user sends

The Send Email To[693] action enables emails to be sent from the client and server. If an email is sent as HTML, you can add a hyperlink to the body of the email. The link can open a MobileTogether solution. To add a link to the email body, use the **`mt-html-anchor`**[1262] function in the XPath expression of the *Body* option (*see screenshot below*).



@ Send Email ○ from Client ● from Server
As ● HTML ○ Text
From "sender@altova.com"
To "contact.one@altova.com"
Cc "contact.two@altova.com; contact.three@altova.com"
Bcc
Subject "MobileTogether Clients Available in EN, DE, FR, ES, JA"
Body concat($XML3/Messages/Message[@date="2015-04-15"], mt-html-anchor('Unregister from mailing list', mt-run-solution-url('', '/public/unregister', '')))
● No Attachments ○ Attachments listed below ○ Dynamic Attachments
On Error ● Abort Script ○ Continue

The **`mt-html-anchor`**[1262] function takes two arguments: **`LinkText`** and **`TargetURL`**. It uses these two arguments to create an HTML hyperlink element: **`<a href="TargetURL">LinkText</a>`**

For example:

```
mt-html-anchor('Unregister from mailing list', mt-run-solution-url('',
'/public/unregister', ''))
```

generates an HTML code fragment of the following pattern:

```
<a href="LinkTo unregister.mtd">Unregister from mailing list</a>
```

The **`mt-run-solution-url`**[1262] function generates the URL that links to the solution (using the `mobiletogether://` scheme), and this URL is stored as the value of the hyperlink's `href` attribute.

**Note:** When a link is created with the `mt-run-solution-url`[1262] function, it is created with the `mobiletogether://` scheme (and not the `http://` scheme), which enables a solution to be opened from the email applications of mobile devices. However, if the email is opened on a web client, the link to open the solution must use the `http://` scheme. In this case, therefore, the `http://` link must be manually created; the `mt-run-solution-url` 1262 function should not be used in this case.

**Note:** For web clients, a link can be created that goes directly to a solution on the server, for example, `http://localhost:8085/run?d=/public/BizBudget`. If the solution's container on the server has been configured to allow anonymous access, then the end user will not need to log in to the server, but can use the solution directly. For information about setting access levels on the server, see the MobileTogether Server user manual.

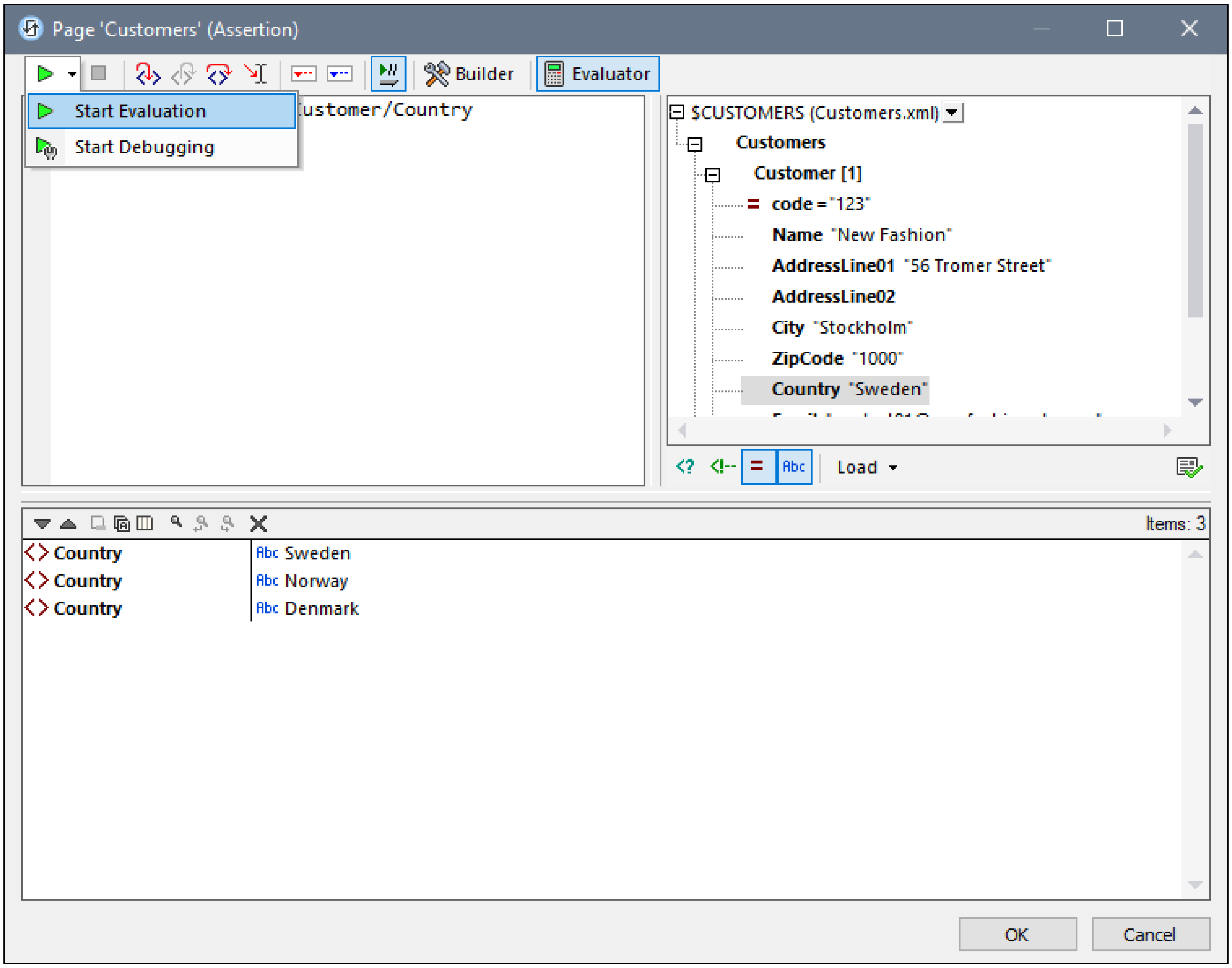
# 13 XPath/XQuery Expressions and Functions

XPath/XQuery expressions play an important role in MobileTogether designs, and MobileTogether Designer provides (i) powerful features to help you edit and debug expressions, as well as (ii) a range of extension functions. This section describes the following:

- XPath/XQuery Window 1244, in which you can build expressions easily, run test evaluations of the expressions, and debug the expressions using a range of diagnostic features
- MobileTogether Extension Functions 1262
- User-Defined XPath/XQuery Functions 1293
- FAQ about XPath/XQuery 1296

# 13.1 XPath/XQuery Window

The **XPath/XQuery Window** (*screenshot below*) is accessed whenever an XPath expression needs to be entered or edited, and from the Simulator[1355]. It is used to correctly create and to test XPath expressions for a range of MobileTogether features.



```
ustomer/Country
```

$CUSTOMERS (Customers.xml)
- Customers
  - Customer [1]
    - = code = "123"
    - Name "New Fashion"
    - AddressLine01 "56 Tromer Street"
    - AddressLine02
    - City "Stockholm"
    - ZipCode "1000"
    - Country "Sweden"

Items: 3

| | |
|---|---|
| <> Country | Abc Sweden |
| <> Country | Abc Norway |
| <> Country | Abc Denmark |

- Expression Builder[1245] provides entry helpers, popup descriptions of context-sensitive XPath/XQuery constructs, and auto-completion. To select Expression Builder, click **Builder** in the toolbar *(see screenshot above)*.
- Expression Evaluator[1248] gives you a preview of expression results, thus enabling you to review and correct your expression. To select Expression Evaluator: Click **Evaluator** in the toolbar, and then select **Start Evaluation** in the dropdown menu of **Start Evaluation/Debugging** *(see screenshot above)*. Alternatively, simply select **Start Evaluation**.
- In XPath Debugger[1252], you can step through the evaluation of an XPath expression to see how the expression is processed. You can subsequently modify the expression as needed. To select XPath Debugger: Click **Evaluator** in the toolbar, and then elect **Start Debugging** in the dropdown menu of **Start Evaluation/Debugging** *(see screenshot above)*. Alternatively, simply select **Start Debugging**.

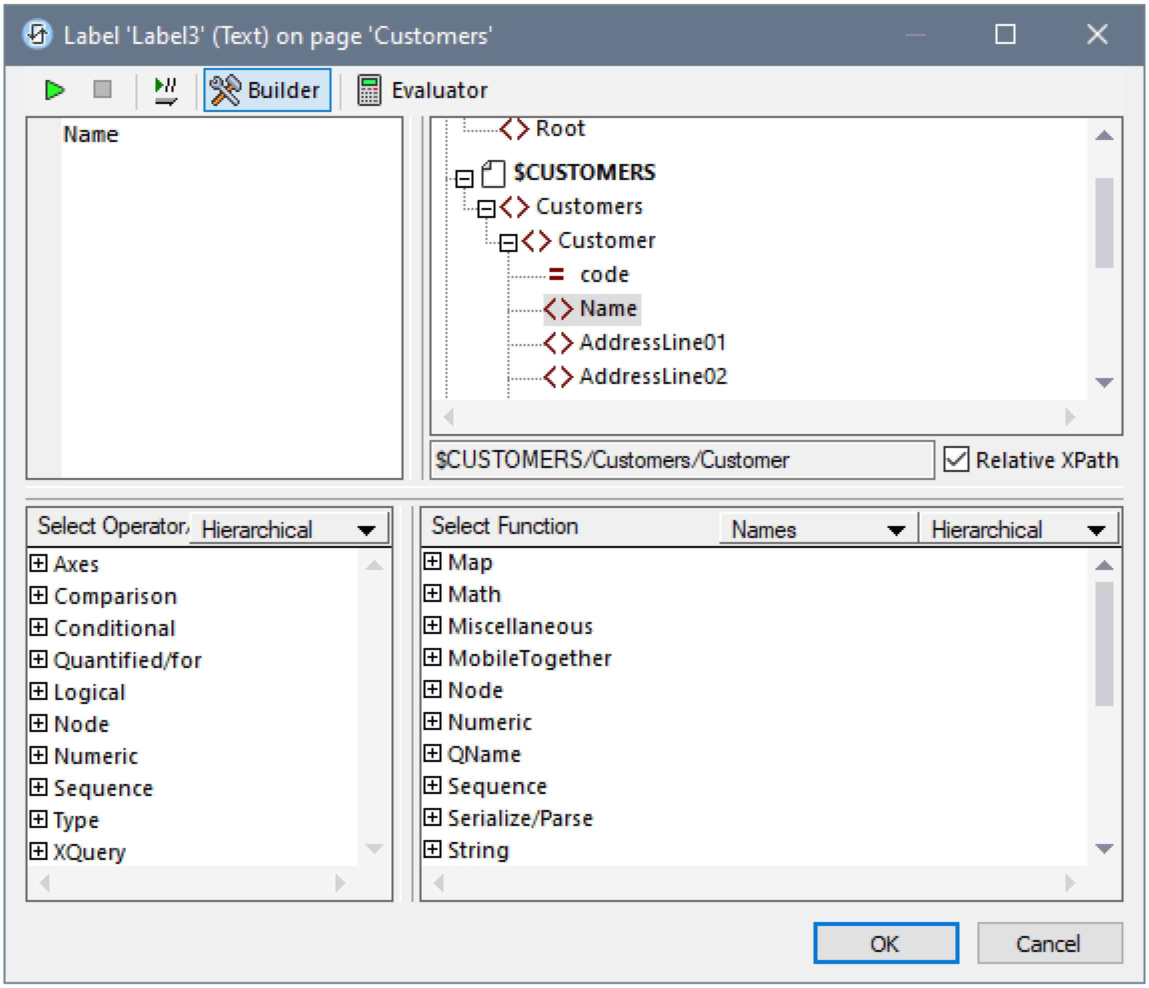
### Availability of the XPath/XQuery Window

The XPath /XQuery Window is available in the following contexts:

- In the design, it is available wherever an XPath expression may be entered, for example, when entering expressions to set the values of styles and properties 274.
- From the simulator 1355. In this context, it is useful for analyzing how the solution will behave in various runtime situations and with different datasets.

## 13.1.1 Expression Builder

The Expression Builder enables you to build correct XPath expressions with respect to the current context node. It enables you to enter correct node locator paths, select nodes in the schema tree, and enter operators and functions that are syntactically correct. Switch to Expression Builder by clicking the **Builder** button (*see screenshot below*). When the **Builder** button is clicked, entry helper panes to help you build an XPath expression become visible. Double-click an entry in the entry helpers to enter it at the current cursor location in the Expression pane. On clicking **OK**, the expression is entered at the current location in the design.



Name

$CUSTOMERS/Customers/Customer

☑ Relative XPath

There are three entry helper panes:

- A schema tree for selecting and entering element and attribute nodes. If the *Relative XPath* check box is checked *(as in the screenshot above)*, then the path to the selected node is entered relative to the context node—which is the node in the design within which the XPath expression is being built. The context node is shown below the schema tree pane. You can see in the screenshot above that the context node is the **`Customer`** element. You can also see, from the title bar of the window, that this XPath expression is being built for the `Text` property of the `Label3` control. If the *Relative XPath* check box is unchecked, then an absolute XPath expression to locate the selected node is entered; such an expression would start at the document root, for example: **`$CUSTOMERS/Customers/Customer/Name`**.
- An entry helper pane for operators and expressions. These include: (i) axes (`ancestor::`, `parent::`, etc), (ii) operators (for example `eq` and `div`), and (iii) expressions (`for # in  # return #`, etc). The items of the pane can be either listed alphabetically or grouped by functional category. Select *Hierarchical* (for the grouped option) or *Flat* (for an alphabetical listing) from the dropdown menu in the title bar of the pane.
- An entry helper with XPath functions either listed alphabetically or grouped by functional category. Select *Hierarchical* (for the grouped option) or *Flat* (for an alphabetical listing) from the dropdown menu in the title bar of the pane. The *Names/Types* option enables you to choose whether the function's

arguments are displayed as names or datatypes. Note that the list of available functions includes MobileTogether extension functions 1262 and Altova extension functions 1689.

*Features of the Builder*

- To view a text description of an item in the Operators pane and Functions pane, hover over the item.
- Each function is listed with its signature (that is, with its arguments, the datatypes of the arguments, and the datatype of the function's output).
- The signatures of functions are given with either the names or the datatypes of the function's arguments and output. Select *Names* or *Types* from the dropdown menu in the title bar of the pane for the respective option.
- Double-click an item in any of the three entry helper panes (operator, expression, or function) to insert that item at the cursor location in the expression. Functions are inserted with their arguments indicated by placeholders (the `#` symbol).
- If (i) text is selected in the XPath expression's edit field (for example `Name` in the screenshot above), and (ii) an expression or function that contains a placeholder is double-clicked to insert it, then the text that was selected is inserted instead of the first placeholder.
- After you have entered a function in the expression, hovering over the function name displays a popup containing the function's signature and a text description of the function. If additional signatures exist for a function, then this is indicated by an overload factor at the bottom of the popup. *(Note: If multiple functions have the same name but each takes a different set of arguments, then, this set of functions is considered, more correctly, to be a single function with multiple signatures.)*
- If, in the Expression pane, you place the cursor within the parentheses of a function and press **Ctrl+Shift+Spacebar**, you can view the different signatures of that function.

## Building XPath expressions

The Edit XPath Expression dialog helps you to build XPath expressions in the following ways.

- *Title bar of window displays*
  The title bar of the XPath/XQuery window displays information about the context (such as the design component (project, page, control, etc) and property) for which the expression is being built. For example, in the screenshot above, the expression is being built for the `Text` property of the `Label3` control on a page named `Customers`.

- *Context node and schema tree*
  The *Context* text box immediately below the Schema Tree pane shows you the context node of the expression. In the schema tree itself, you can see where the context node occurs and quickly build the XPath expression by referring to the schema tree.

- *Inserting a node from the schema tree*
  Double-click a node in the schema tree to insert it in the XPath expression. If the *Relative XPath* check box is checked, then the node is inserted with a location path that is relative to the context node. For example, in the screenshot above, the `Name` element, which is a child of the `Customer` element (the context node), has been inserted with a path that is relative to the `Customer` element. If the *Relative XPath* check box is not checked, then the `Name` node would be inserted as `$CUSTOMERS/Customers/Customer/Name`.

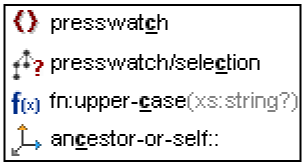
- *Inserting XPath axes, operators and expressions*
  The *Select Operator/Expression* pane lists the XPath axes (`ancestor::`, `parent::`, etc), operators (for example, `eq` and `div`), and expressions (`for # in  # return #`, etc). The display can be toggled between an alphabetical listing and a hierarchical listing (which groups the items according to

functionality). To insert an axis or operator in the XPath expression, double-click the required item. Placing the mouse cursor over an axis/operator/expression displays a brief description of that item.

- *Inserting XPath functions*
  The *Select Function* pane lists XPath functions alphabetically or grouped according to functionality (select *Hierarchical* or *Flat* at the top of the pane to switch between the two arrangements). Each function is listed with its signature. If a function has more than one signature, that function is listed as many times as the number of signatures. Arguments in a signature are separated by commas, and each argument can have an occurrence indicator. The symbol **?** indicates a sequence of zero or one items of the specified type; ***** indicates a sequence of zero or more items of the specified type; **+** indicates a sequence of one or more items of the specified type. The arguments can be displayed as names or as datatypes. Select the appropriate option (*Names* or *Types*) at the top of the pane to toggle between the two display options. Each function also specifies the return type of that function. For example: `=> date ?` indicates that the expected return datatype is a sequence of zero or one `date` items. Placing the mouse cursor over a function displays a brief description of the function.

## Intelligent editing during direct text entry

If you type an expression directly in the *Expression* text box, a list of options that are available at that point are displayed in a popup (*see screenshot below*).



These include the following components:

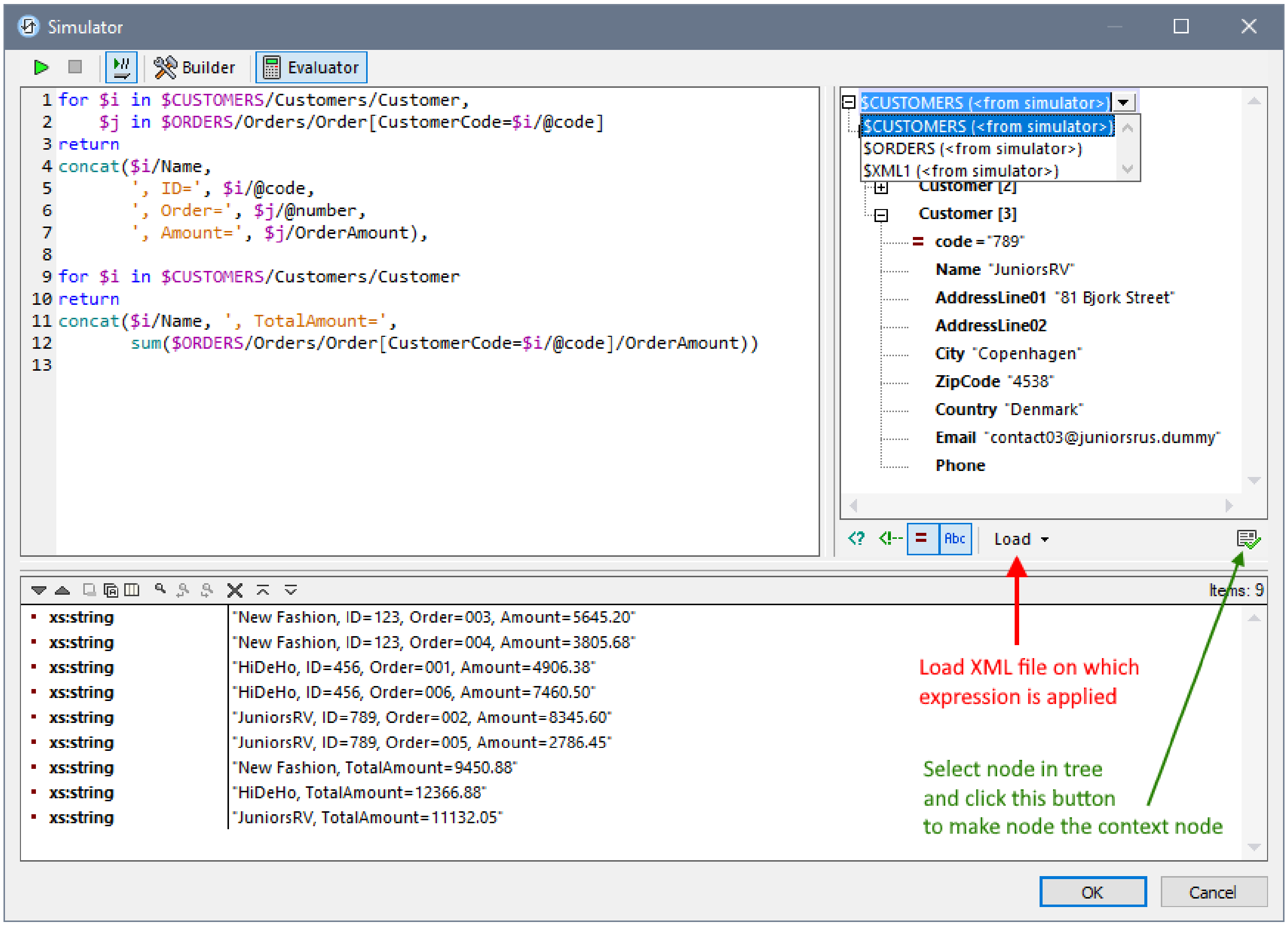
- elements (such as `presswatch` in the screenshot above)
- descendant nodes (`presswatch/selection` in the screenshot above), and parent node of the context node
- XPath functions (`fn:upper-case` above) and XPath axes (`ancestor-or-self` above)
- a list of the global variables 1298 defined for the project (displayed when a `$` character is entered in the expression)
- a list of the custom strings 1600 defined in the Localization dialog (displayed when the `mt-load-string` 1262 function is entered in the expression; see the description of `mt-load-string` 1262)

Go up and down the list of options using the **Up** and **Down** keys, and press **Enter** if you wish to select an option and enter it in the expression.

## 13.1.2 Expression Evaluator

The Expression Evaluator *(screenshot below)* enables you to apply an XPath/Query expression to a data source file and preview the return value. In Expression Evaluator, the window has three panes:

- an *Expression* pane (at top left), in which you enter the XPath/XQuery expression to evaluate
- a *Schema Tree* pane (top right), which shows the structure of the document that is currently loaded; the pane also acts as an entry helper for adding nodes from schema trees
- a *Results* pane (bottom), which displays the result of the evaluation


Simulator
Builder
Evaluator
$CUSTOMERS (<from simulator>)
$CUSTOMERS (<from simulator>)
$ORDERS (<from simulator>)
$XML1 (<from simulator>)
Load
Items: 9
OK
Cancel

```
for $i in $CUSTOMERS/Customers/Customer,
    $j in $ORDERS/Orders/Order[CustomerCode=$i/@code]
return
concat($i/Name,
       ', ID=', $i/@code,
       ', Order=', $j/@number,
       ', Amount=', $j/OrderAmount),

for $i in $CUSTOMERS/Customers/Customer
return
concat($i/Name, ', TotalAmount=',
       sum($ORDERS/Orders/Order[CustomerCode=$i/@code]/OrderAmount))
```

```
Customer [2]
Customer [3]
    = code = "789"
    Name "JuniorsRV"
    AddressLine01 "81 Bjork Street"
    AddressLine02
    City "Copenhagen"
    ZipCode "4538"
    Country "Denmark"
    Email "contact03@juniorsrus.dummy"
    Phone
```

```
xs:string   "New Fashion, ID=123, Order=003, Amount=5645.20"
xs:string   "New Fashion, ID=123, Order=004, Amount=3805.68"
xs:string   "HiDeHo, ID=456, Order=001, Amount=4906.38"
xs:string   "HiDeHo, ID=456, Order=006, Amount=7460.50"
xs:string   "JuniorsRV, ID=789, Order=002, Amount=8345.60"
xs:string   "JuniorsRV, ID=789, Order=005, Amount=2786.45"
xs:string   "New Fashion, TotalAmount=9450.88"
xs:string   "HiDeHo, TotalAmount=12366.88"
xs:string   "JuniorsRV, TotalAmount=11132.05"
```

Load XML file on which expression is applied

Select node in tree and click this button to make node the context node

The XPath expression in the screenshot above is applied to the tutorial file **SubpagesAndVisibility.mtd** 188. You can open this design and try out the XPath expression (listed below) during a simulation 1355 (started with **F5**),

⊟ *Listing of XPath expression shown in screenshot above*

```
for $i in $CUSTOMERS/Customers/Customer,
       $j in $ORDERS/Orders/Order[CustomerCode=$i/@code]
return
concat($i/Name,
       ', ID=', $i/@code,
       ', Order=', $j/@number,
       ', Amount=', $j/OrderAmount),

for $i in $CUSTOMERS/Customers/Customer
return
concat($i/Name, ', TotalAmount=',
       sum($ORDERS/Orders/Order[CustomerCode=$i/@code]/OrderAmount))
```

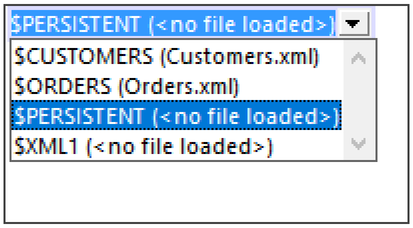
## Using Expression Builder and switching to Expression Evaluator for the results

- You can switch between the Builder and the Evaluator without losing the expression that was typed in the Expression pane. To switch, click the appropriate toolbar button (**Builder** or **Evaluator**).
- The advantages of the Builder are entry helper support and the simultaneous availability of schema trees of all page sources. (In the Evaluator, on the other hand, only one schema tree is available at a time.)
- After building an expression, you can evaluate it in Expression Evaluator by selecting **Start Evaluation**.
- If the **Evaluate on Typing** button in the window's toolbar is toggled on, then the expression is evaluated as you type in Expression Evaluator.

## Data source on which the expression is applied

Two situations may be distinguished in this context:

- If the XPath/XQuery Window is opened from the Simulator window 1355, then the data sources of the design have already been loaded for the simulation. As a result, evaluation can be directly carried out on these data sources. You can see the structures (schema trees) of these data sources in the Schema Tree pane of both the Evaluator and Builder. (In Expression Evaluator, however, only one schema tree is displayed at a time *(see screenshot above)*. To switch trees in the Schema Tree pane of Expression Evaluator, select another tree in the dropdown menu of the schema tree selector *(see screenshot above)*.
- If the XPath/XQuery Window is opened from within the design and no corresponding XML file is loaded, this is indicated *(see screenshot below).* In this case, you must explicitly load an XML file that has the same structure as one of the page sources of the design. Do this via the **Load** button below the Schema Tree pane *(see screenshot above that shows Schema Tree)*.


$PERSISTENT (<no file loaded>)
$CUSTOMERS (Customers.xml)
$ORDERS (Orders.xml)
$PERSISTENT (<no file loaded>)
$XML1 (<no file loaded>)

## Schema Tree pane

In addition to displaying the selected schema tree (described above), the Schema Tree pane provides the following features:

- The icons below the pane toggle on/off the display of the following XML syntactic constructs: (i) processing instructions, (ii) comments, (iii) attributes, (iv) text nodes. You can therefore see the entire XML document structure, together with the text contents of nodes, but you can also hide certain constructs if you wish to reduce clutter in the pane.
- You can change the context node of the XPath expression by clicking the node in the document tree that you want as the new context node and then selecting **Set Evaluation Context**. (located below the Schema Tree pane at right). Note, however, that the actual context node for the expression at run time will be the context node within which the current design component is being created.

## Expression pane

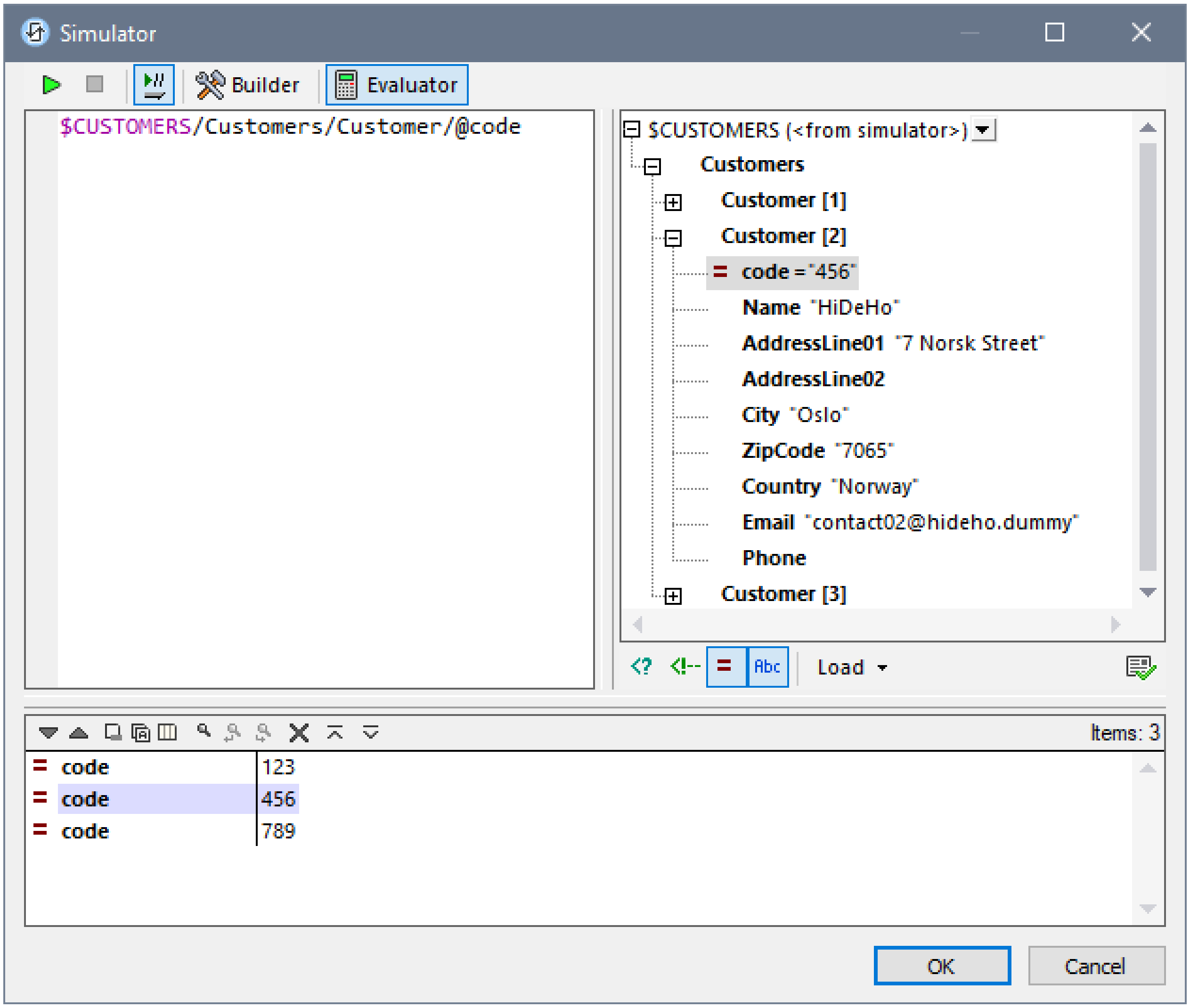
The XPath/XQuery expression is entered in the Expression pane. The results of the evaluation are displayed in the *Results* pane *(see screenshot above)*.

Note the following points:

- To create the expression over multiple lines (for easier readability), use the **Return** key.
- To increase/decrease the size of text in the expression field, click in the expression field, then press **Ctrl** and turn the scroll wheel. ***Note that this also applies in the Results pane.***

## Results pane

The Results pane is shown in the screenshot below, at bottom. Note that it has its own toolbar.



```
$CUSTOMERS/Customers/Customer/@code
```

$CUSTOMERS (<from simulator>)
Customers
Customer [1]
Customer [2]
code = "456"
Name "HiDeHo"
AddressLine01 "7 Norsk Street"
AddressLine02
City "Oslo"
ZipCode "7065"
Country "Norway"
Email "contact02@hideho.dummy"
Phone
Customer [3]

Items: 3

| | |
|---|---|
| code | 123 |
| code | 456 |
| code | 789 |

The Results pane has the following functionality:

- The result list consists of two columns: (i) a node name or a datatype; (ii) the content of the node.
- If the XPath expression returns nodes (such as elements or attributes), you can select whether the entire contents of the nodes should be shown as the value of the node. To do this, switch on the toggle *Show Complete Result*.
- For data sources loaded during simulations: When the result contains a node (including a text node)—as opposed to expression-generated literals—clicking that node in the Results pane highlights the corresponding node in the Schema Tree pane.
- You can copy both columns of a result sub-line, or only the value column. To copy all columns, right-click a sub-line and toggle on **Copying Includes All Columns**. (Alternatively you can toggle the command on/off via its icon in the toolbar of the Results pane.) Then right-click the sub-line you want to copy and select either **Copy Subline** (for that subline) or **Copy All** (for all sublines).

*Toolbar of the Results pane*
The toolbar of the Results pane contains icons that provide navigation, search, and copy functionality. These icons, starting from the left, are described in the table below. The corresponding commands are also available in the context menu of result list items.

| Icon | What it does |
|---|---|
| *Next, Previous* | Selects, respectively, the next and previous item in the result list |
| *Copy the selected text line to the clipboard* | Copies the value column of the selected result item to the clipboard. To copy all columns, toggle on the *Copying includes all columns* command *(see below)* |
| *Copy all messages to the clipboard* | Copies the value column of all result items to the clipboard, including empty values. Each item is copied as a separate line |
| *Copying includes all columns* | Switches between copying (i) all columns, or (ii) only the value column. The column separator is a single space |
| *Find* | Opens a *Find* dialog to search for any string, including special characters, in the result list |
| *Find previous* | Finds the previous occurrence of the term that was last entered in the *Find* dialog |
| *Find next* | Finds the next occurrence of the term that was last entered in the *Find* dialog |
| *Expand with children* | Expands the selected item and all its descendants |
| Collapse with children | Collapses the selected item and all its descendants |
| *Clear* | Clears the result list |

## 13.1.3 XPath Debugger

The XPath Debugger of the XPath/XQuery Window enables you to debug XPath expressions in the context of the loaded file/s. You can open the XPath/XQuery Window via the following entry points:

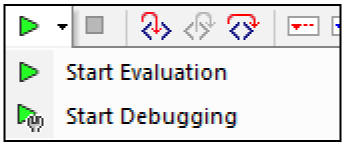
- In the design, it is opened wherever an XPath expression may be entered or edited: for example, when entering expressions to set the values of styles and properties 274.
- From the simulator 1355 it is opened by clicking the **Evaluate XPath** 1398 button in the Page Sources pane.
- When the simulator 1355 stops at an action, you can start XPath Debugger by clicking the **Step Into XPath** 1398 button of the Actions Debugger 1390.

**Note:** In the first two contexts listed above, you can enter any XPath expression you like and debug it against any XML file, which you must load into the Debugger. In the third context, however, you will be debugging the XPath expression that is specific to the current action, and you will be debugging this expression against the current action's page sources; consequently, the option to load an XML file is disabled.

The descriptions and screenshots in this section refer to the XPath Debugger when accessed via the Simulator's Page Sources pane[1398]. The Debugger behaves in the same way irrespective of the context from which it was launched. Note, however, that the option to load an XML file is disabled when the Debugger is started from the third context.

## Starting XPath Debugger

To start XPath Debugger, select **Start Debugging** in the **Start Evaluation/Debugging** dropdown menu *(screenshot below)*. You can switch between the Builder (for help with building the expression) and the Debugger.



Either before or after entering an XPath/XQuery expression, select an XML file on which the expression is to run (see *Running the Debugger*[1255] below). Start debugging by clicking **Start Evaluation/Debugging (F5)**. The Debugger will run the expression on the loaded XML file and display results in the panes in the lower part of the window.

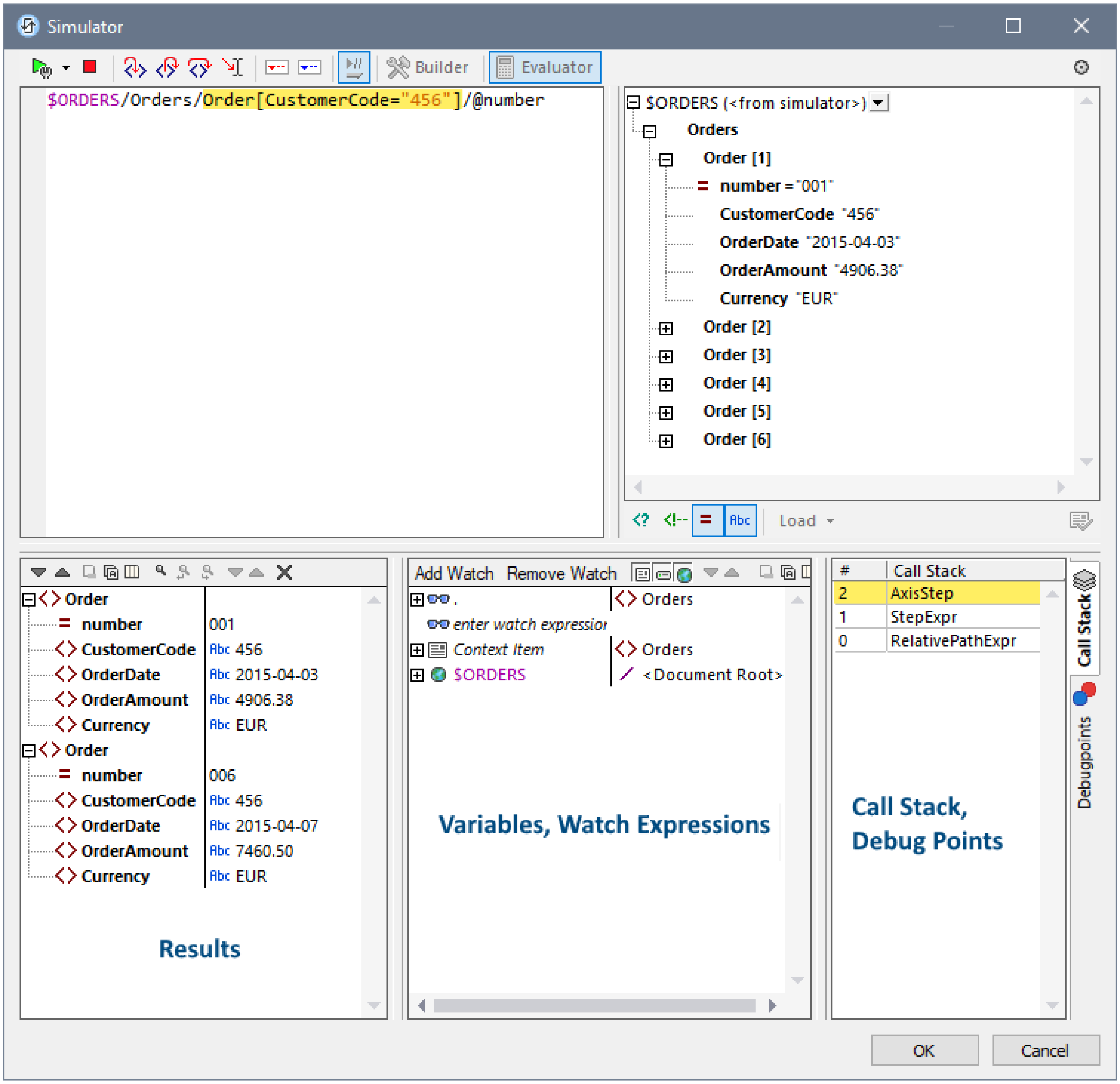
⊟ *Buttons for setting up XPath Debugger*

| | | |
|---|---|---|
| | **Start Evaluation/Debugging (F5)** | Starts the debugger |
| | **Switch to Builder** | Switches to Expression Builder mode, which provides context-sensitive entry helpers to help construct expressions |
| | **Evaluation on typing** | Switches on the evaluation of expressions while the expression is being typed |

## Layout of XPath Debugger

XPath Debugger has two panes that are additional to the Results pane of the Evaluator *(see screenshot below)*:

- the Watch Expressions and Variables pane; both, variables and watch expressions, are shown in the same pane
- the Call Stack and Debug Points pane, each of which has a separate tab on the right-hand side of the pane



```
$ORDERS/Orders/Order[CustomerCode="456"]/@number
```

XPath Debugger provides the following features:

- Enables you to step into the XPath evaluation process, one step at a time to see how the XPath expression is being evaluated. Use the **Step Into (F11)** toolbar button for this. At each evaluation step, the part of the expression being currently evaluated is highlighted in yellow *(see screenshot above)*, while the result of evaluating that step is shown in the Results pane. For example, in the screenshot above, all the `Order` elements that have a descendant element named `CustomerCode` containing the value `'456'` have been selected.
- Set breakpoints where you want to pause the evaluation and check results. Use the **Start Debugging (F5)** toolbar button to go through the evaluation, pausing only at breakpoints. Using breakpoints is quicker than pausing at every processing step with **Step Into (F11)**.

- Set tracepoints to see the evaluation results at the steps marked as tracepoints. The evaluation will not pause—they pause only at breakpoints—but the tracepoint results will be displayed in a list in the Results pane.
- Watch expressions are XPath expressions that are evaluated at every step of debugging. You can use them to check information you want at each step, such as document data or aspects of the evaluation. Watch expressions are very useful when used together with breakpoints.
- Variables that are in scope at the current step, including their values, are displayed in the Watch Expressions and Variables pane.
- Processor calls of an evaluation step are shown in the Call Stack tab of the Call Stack and Debug Points pane.
- If breakpoints and tracepoints have been set, then these are displayed in the Debug Points tab of the Call Stack and Debug Points pane.

For more information about these features, see their descriptions below.

## Running the Debugger

The broad steps for debugging an XPath expression are as follows:

1. Enter the XPath expression in the expression pane.
2. Make sure that the XML file on which you want to apply the XPath expression is loaded. If it is not loaded, then use the **Load** button (located below the Schema Tree pane; *see screenshot above*) to browse for the file and load it.
3. Set any breakpoints or tracepoints you want. A breakpoint is a point at which the evaluation is paused. A tracepoint is a point in the evaluation at which the result is recorded; tracepoints thus provide a traceable path of evaluation results.
4. If you click **Start Debugger**, evaluation is paused at the first breakpoint. Click **Start Debugger** repeatedly to progress through each breakpoint to the end of the evaluation.
5. Use the Step Into/Out/Over functionality to go step-by-step (as opposed to breakpoint-by-breakpoint) through the evaluation.
6. You can also run the debugger up to a point in the XPath expression: (i) Mark the point by placing your cursor at that location, and (ii) click the **Run to Cursor** toolbar icon.

⊟ *Buttons for debugging*

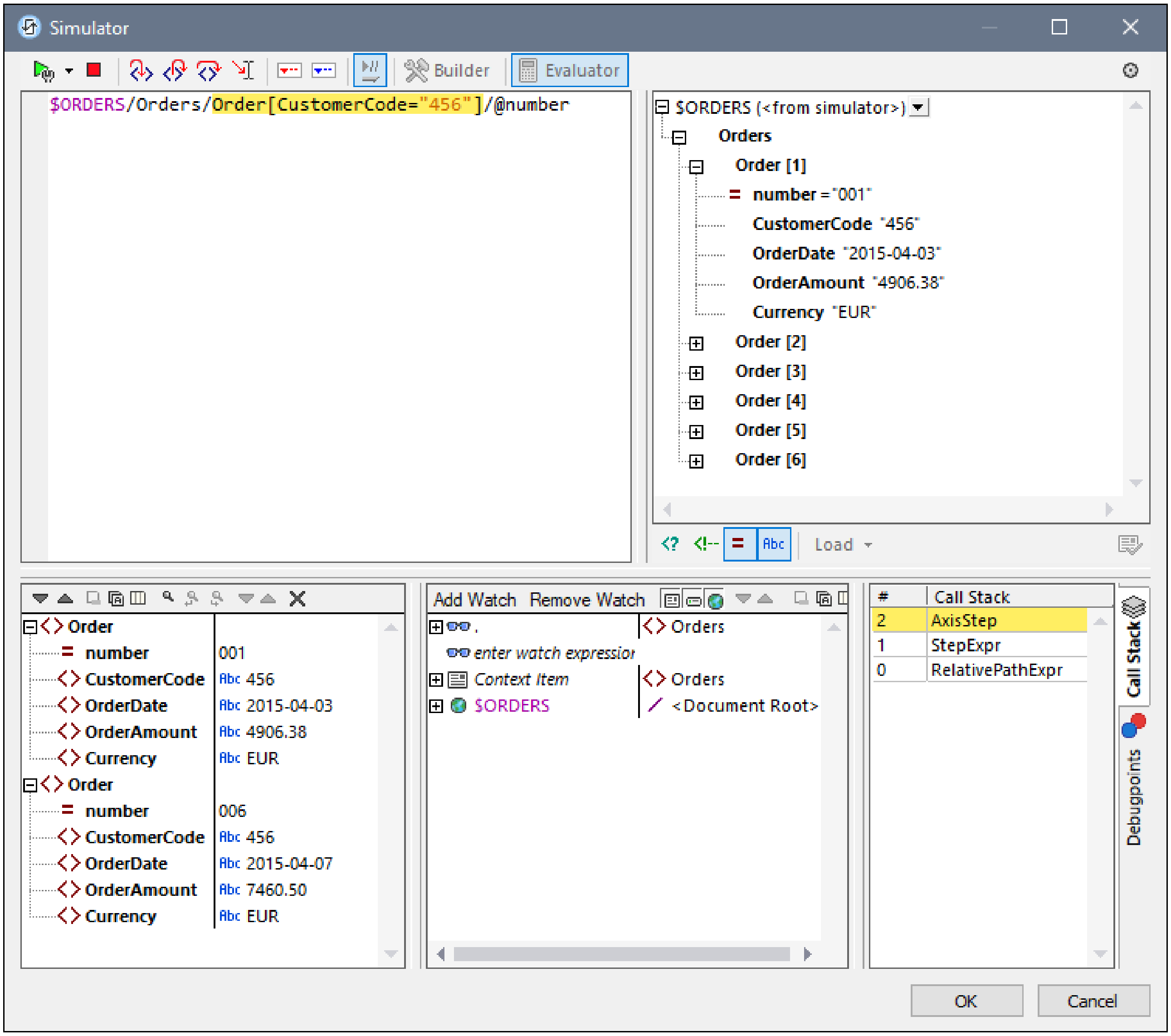
| | | |
|---|---|---|
| ▶ | **Start Debugger (F5)** | Starts the debugger. Evaluation pauses at the next breakpoint |
| 🟥 | **Stop Debugger (Shift+F5)** | Exits the evaluation and stops the debugger |
| | **Step Into (F11)** | Proceeds through the evaluation, one step at a time. |
| | **Step Out (Shift+F11)** | Steps out of the current evaluation step, and goes to the parent step |
| | **Step Over (Ctrl+F11)** | Steps over descendant steps |
| | **Run to Cursor (Ctrl+F5)** | Starts the debugger and runs it to the cursor location in the XPath expression |
| | **Insert/Remove Breakpoint (F9)** | Inserts/removes a breakpoint at the expression step where you place the cursor |

| | | |
|---|---|---|
| | **Insert/Remove Tracepoint (Shift+F9)** | Inserts/removes a tracepoint at the expression step where you place the cursor |

*Stepping into, out of, and over evaluation steps*

The *Step Into* functionality enables you to go step-by-step through the evaluation. Each click of this command takes you through the next step of the evaluation; the current step is shown by the highlighting in the expression *(see screenshot below)*. The *Step Out* functionality takes you to a step on a higher level as the current step, whereas the *Step Over* functionality steps over lower-level steps and takes you to the next step on the same level. You can try out the *Stepping* functionality by using the expression shown in the screenshot below (**`$ORDERS/Orders/Order[CustomerCode="456"]/@number`**) and clicking the three *Step* buttons to see how they work. (The XML file in this example is a page source of the tutorial SubPages-And-Visibility[188]. Open this file and do the following: (i) Start the Simulator; (ii) On the top page that appears in the Simulator, click *Show All Orders*, (iii) In the Page Sources pane of the Simulator, click the Evaluate XPath toolbar icon, (iv) switch to the Debugger *(see above)*.)

The screenshot below shows the evaluation when processing has been paused on reaching the locator step **`Order[CustomerCode='456']`**. At this step, the result shows the two **`Order`** elements that each have a `CustomerCode` child element containing the value `'456'`. The two elements (with their entire node contents) are displayed in the Results pane.



```
$ORDERS/Orders/Order[CustomerCode="456"]/@number
```

```
$ORDERS (<from simulator>)
  Orders
    Order [1]
      = number ="001"
      CustomerCode "456"
      OrderDate "2015-04-03"
      OrderAmount "4906.38"
      Currency "EUR"
    Order [2]
    Order [3]
    Order [4]
    Order [5]
    Order [6]
```

| | |
|---|---|
| Order | |
| = number | 001 |
| CustomerCode | Abc 456 |
| OrderDate | Abc 2015-04-03 |
| OrderAmount | Abc 4906.38 |
| Currency | Abc EUR |
| Order | |
| = number | 006 |
| CustomerCode | Abc 456 |
| OrderDate | Abc 2015-04-07 |
| OrderAmount | Abc 7460.50 |
| Currency | Abc EUR |

| | |
|---|---|
| . | Orders |
| *enter watch expression* | |
| *Context Item* | Orders |
| $ORDERS | <Document Root> |

| # | Call Stack |
|---|---|
| 2 | AxisStep |
| 1 | StepExpr |
| 0 | RelativePathExpr |

## Breakpoints

Breakpoints are points where you want the Debugger to stop after it has been started with **Start Debugger**. They are useful if you wish to analyze a specific part of the expression. When the Debugger stops at the breakpoint, you can check the result and could then use the **Step Into** functionality to display the results of the next steps of the evaluation. To set a breakpoint, place the cursor in the expression at the point where you want the breakpoint, and click the **Insert/Remove Breakpoint (F9)** toolbar button. The breakpoint will be marked with a dashed red overline. To remove a breakpoint, select it and click **Insert/Remove Breakpoint (F9)**.

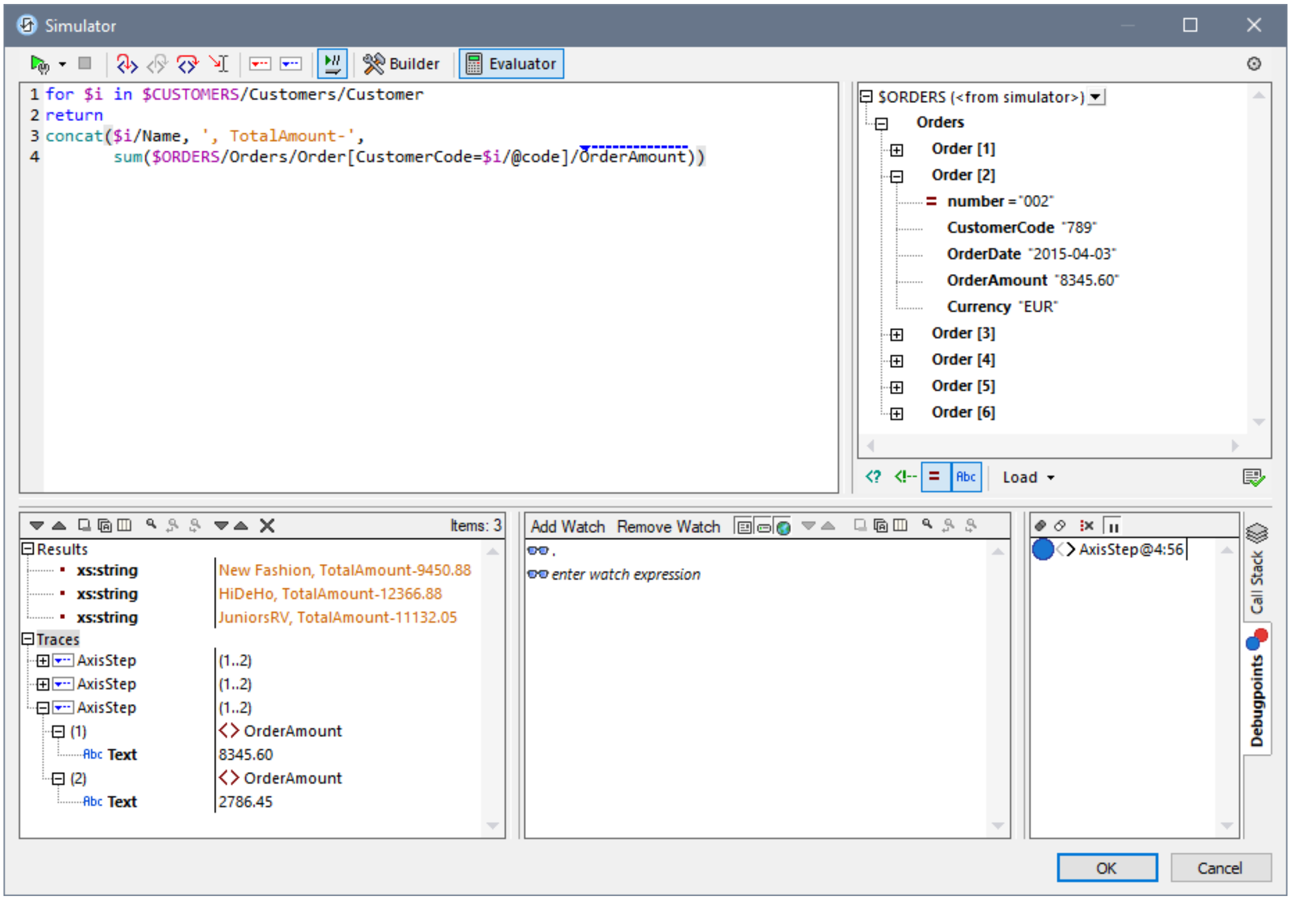
**Note:** You can make a breakpoint conditional by entering the condition on the breakpoint's listing in the Debug Points pane. See Debug Points 1260 below.

**Note:** You can start XPath Debugger directly from a simulation to debug XPath expressions that have breakpoints set on them. To do this start MT Debugger 1388 in Breakpoints Mode 1389.

## Tracepoints

Tracepoints are points at which the evaluation results are recorded. These results are displayed in the *Traces* tree of the Results pane (*see screenshot below*). Tracepoints enable you to see the evaluation results of particular parts of the expression. For example, in the screenshot below, tracepoints have been set on the `OrderAmount` node. The results at these tracepoints are shown in the *Traces* tree.

To set a tracepoint, place the cursor at the point where you want the tracepoint, and click the toolbar button **Insert/Remove Tracepoint (Shift+F9)**. The tracepoint will be marked with a dashed blue overline (*see screenshot below*). To remove a tracepoint, select it and click **Insert/Remove Tracepoint (F9)**.



```
for $i in $CUSTOMERS/Customers/Customer
return
concat($i/Name, ', TotalAmount-',
       sum($ORDERS/Orders/Order[CustomerCode=$i/@code]/OrderAmount))
```

```
$ORDERS (<from simulator>)
  Orders
    Order [1]
    Order [2]
      = number = "002"
      CustomerCode "789"
      OrderDate "2015-04-03"
      OrderAmount "8345.60"
      Currency "EUR"
    Order [3]
    Order [4]
    Order [5]
    Order [6]
```

```
Results
  xs:string      New Fashion, TotalAmount-9450.88
  xs:string      HiDeHo, TotalAmount-12366.88
  xs:string      JuniorsRV, TotalAmount-11132.05
Traces
  AxisStep       (1..2)
  AxisStep       (1..2)
  AxisStep       (1..2)
    (1)          <> OrderAmount
      Abc Text   8345.60
    (2)          <> OrderAmount
      Abc Text   2786.45
```

```
enter watch expression
```

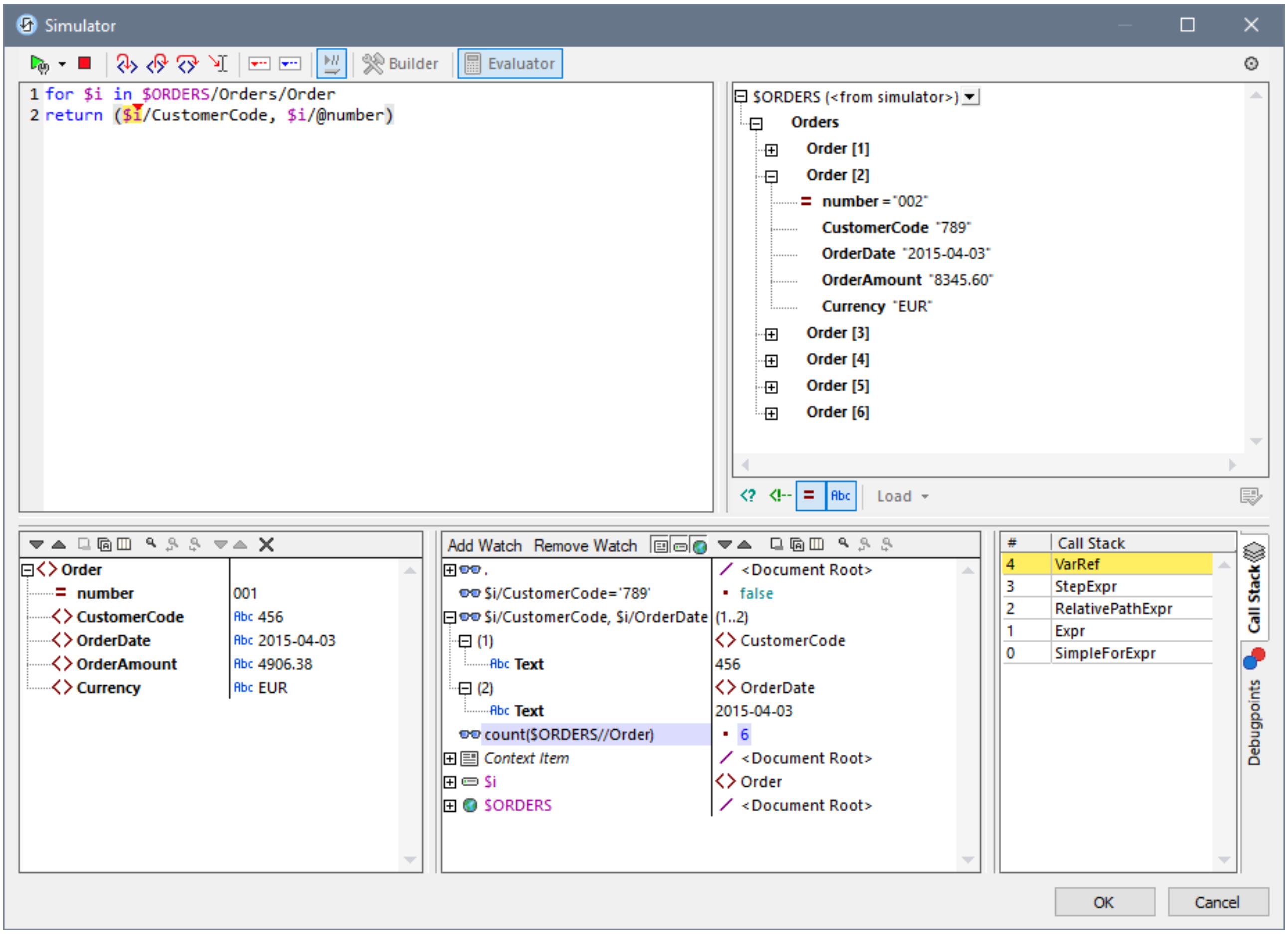
```
AxisStep@4:56
```

**Note:** If both a breakpoint and a tracepoint are set on the same part of an expression, then the overline is composed of alternating red and blue dashes.

Also see Debug Points 1260 below.

## Watch Expressions, Variables and Call Stack

Watch expressions and variables are displayed in the Watch Expressions and Variables pane *(bottom center pane in the screenshot below)*. In this pane, you can filter the view to show/hide (i) the current context item, (ii) local variables, (iii) global variables. Do this by toggling on/off the respective toolbar icons. The current context item is the node that is currently being evaluated.



```
for $i in $ORDERS/Orders/Order
return ($i/CustomerCode, $i/@number)
```

```
$ORDERS (<from simulator>)
  Orders
    Order [1]
    Order [2]
      number = "002"
      CustomerCode "789"
      OrderDate "2015-04-03"
      OrderAmount "8345.60"
      Currency "EUR"
    Order [3]
    Order [4]
    Order [5]
    Order [6]
```

| | |
|---|---|
| Order | |
| number | 001 |
| CustomerCode | 456 |
| OrderDate | 2015-04-03 |
| OrderAmount | 4906.38 |
| Currency | EUR |

| | |
|---|---|
| . | <Document Root> |
| $i/CustomerCode='789' | false |
| $i/CustomerCode, $i/OrderDate | (1..2) |
| (1) | CustomerCode |
| Text | 456 |
| (2) | OrderDate |
| Text | 2015-04-03 |
| count($ORDERS//Order) | 6 |
| *Context Item* | <Document Root> |
| $i | Order |
| $ORDERS | <Document Root> |

| # | Call Stack |
|---|---|
| 4 | VarRef |
| 3 | StepExpr |
| 2 | RelativePathExpr |
| 1 | Expr |
| 0 | SimpleForExpr |

## *Watch expressions*

Watch expressions are XPath expressions that you can enter, either before evaluation starts or during a pause in evaluation. The expression is evaluated in the context of the current node at each evaluation step where the debugger stops. Watch expressions can be used for the following purposes:

- To test specific conditions. For example in the screenshot above, the watch expression `$i/CustomerCode="789"` is used to test whether the current **`Order`** element has a **`CustomerCode`** with a value of **`"789"`**. The result **`false`** in the case of the first **`Order`** element tells us that this order does not have a **`CustomerCode`** value of **`"789"`**. (The value is **`"456"`** as can be seen in the Results pane.)
- To find data within a certain context. For example, in the screenshot above, we have entered the watch expression `$i/CustomerCode, $i/OrderDate` to look up these details of the current order.
- To generate additional data. For example, in the screenshot above, we have entered the watch expression `count($ORDERS//Order)` to count all the **`Order`** elements.

To enter a watch expression, click **Add Watch** in the pane's toolbar, then double-click the new watch entry to enter the XPath expression and click **Enter** when done. To remove a watch expression, select it and click **Remove Watch** in the toolbar. If, during debugging, the expression cannot be correctly evaluated for some reason (for example, if one of its variables is out of scope), then the watch expression turns red.

*Variables*

Variables that have been declared in the expression and that are in scope in the current evaluation step will be displayed together with their respective current values. They are displayed below the watch expressions.

In the screenshot above, for example, processing has been paused at the breakpoint on the variable **`$i`**. The **`$i`** variable is in scope at this evaluation step. So **`$i`** is displayed with its current value, which in the screenshot above is the first **`Order`** element. Since the document's root node, **`$ORDERS`**, is also a variable, it is also displayed with its content, which is the document root.

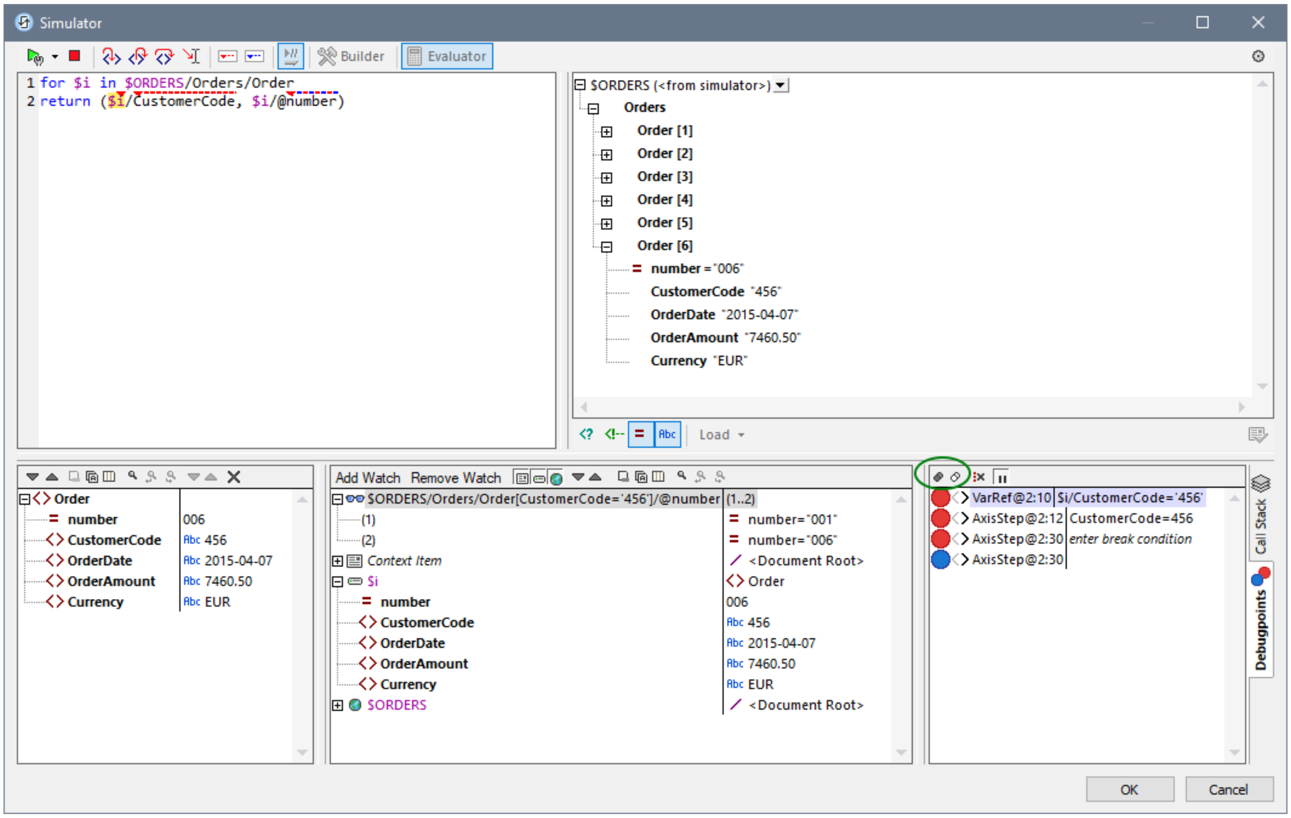
*Call stack*

The *Call Stack* tab of the Call Stack and Debug Points pane *(bottom right pane in the screenshot above)* displays the processor calls up to that point in the debugging. The current processor call is highlighted in yellow. Note that only the calls that directly led to the current evaluation step are displayed.

## Debug Points

The Debug Points tab of the Call Stack and Debug Points pane *(bottom right pane in the screenshot below)* shows the breakpoints (with solid red circles) and tracepoints (solid blue circles) that you have set on the expression. Each debug point is listed with its line and character number. For example, `AxisStep@2:12` means that there is a debug point on line 2, character 12 of the expression in the expression pane.

To stop at each breakpoint in the expression, click **Start Debugger (F5)**.



```
for $i in $ORDERS/Orders/Order
return ($i/CustomerCode, $i/@number)
```

```
$ORDERS (<from simulator>)
  Orders
    Order [1]
    Order [2]
    Order [3]
    Order [4]
    Order [5]
    Order [6]
      number="006"
      CustomerCode "456"
      OrderDate "2015-04-07"
      OrderAmount "7460.50"
      Currency "EUR"
```

```
Order
  number          006
  CustomerCode    456
  OrderDate       2015-04-07
  OrderAmount     7460.50
  Currency        EUR
```

```
$ORDERS/Orders/Order[CustomerCode='456']/@number   (1..2)
  (1)                                               number="001"
  (2)                                               number="006"
Context Item                                        <Document Root>
$i                                                  Order
  number                                            006
  CustomerCode                                      456
  OrderDate                                         2015-04-07
  OrderAmount                                       7460.50
  Currency                                          EUR
$ORDERS                                             <Document Root>
```

```
● VarRef@2:10    $i/CustomerCode='456'
● AxisStep@2:12  CustomerCode=456
● AxisStep@2:30  enter break condition
● AxisStep@2:30
```

Note the following features:

- You can make each breakpoint conditional on some factor by entering a **break condition** for it. Do this in the Debug Points pane by (i) double-clicking the breakpoint's *Enter break condition* entry, (ii) entering the expression for the condition, and (iii) pressing **Enter**. That breakpoint will be enabled only when the condition evaluates to **`true`**. For example, in the screenshot above, the break condition `$i/CustomerCode="456"` will enable the breakpoint on **`$i`** (that is, the current **`Order`** element) when the **`Order`** element's **`CustomerCode`** child element has a value of **`"456"`**. In effect, processing will pause only at those orders that have a customer code of **`456`**. The screenshot shows the evaluation paused at the sixth order, which is such an order. The breakpoint is not triggered for orders with other customer codes. (You can set a watch expression, as in the screenshot above, to show which orders have a customer code of **`456`**: the orders with the numbers **`001`** and **`006`**.)
- You can enable/disable all debug points by clicking their respective toolbar buttons: **Enable All Debug Points** and **Disable All Debug Points** *(buttons encircled in green in the screenshot above)*. When a debug point is disabled, it is deactivated for all evaluations till it is enabled again.
- You can enable/disable individual breakpoints in their respective context menus.
- Tracepoints are simply listed with their details. They can be enabled/disabled.
- If you click **Clear All Debug Points**, then all breakpoints and tracepoints in the design will be removed.
- If an error occurs, for which no handling has been defined, this can be picked up. Simply switch on the **Break on Unhandled Error** toggle command in the toolbar. The Debugger will break on such errors.

## Toolbar commands in panes

The panes of the XPath/XQuery Window in Debug Mode (*see screenshot above*) contain buttons that provide navigation, search, and copy functionality. These buttons, starting from the leftmost in each pane, are described in the table below. The corresponding commands are also available in the context menu of listed items.

| Icon | What it does |
|---|---|
| *Next, Previous* | Selects, respectively, the next and previous item in the result list |
| *Copy the selected text line to the clipboard* | Copies the value column of the selected result item to the clipboard. To copy all columns, toggle on the *Copying includes all columns* command *(see below)* |
| *Copy all messages to the clipboard* | Copies the value column of all result items to the clipboard, including empty values. Each item is copied as a separate line |
| *Copying includes all columns* | Switches between copying (i) all columns, or (ii) only the value column. The column separator is a single space |
| *Find* | Opens a *Find* dialog to search for any string, including special characters, in the result list |
| *Find previous* | Finds the previous occurrence of the term that was last entered in the *Find* dialog |
| *Find next* | Finds the next occurrence of the term that was last entered in the *Find* dialog |
| *Clear* | Clears the result list |

## Closing the XPath Debugger

To close the XPath Debugger, click **Stop Evaluation/Debugging**.

## 13.2 MobileTogether Extension Functions

The following XPath extension functions, created specifically for use in MobileTogether designs, can be used in XPath expressions anywhere in the design. The XPath default namespace is used for calls to these extension functions.

### Updating the return values of XPath functions

An XPath function is evaluated only when its containing XPath expression is evaluated. This typically happens when an action that contains the XPath expression is triggered or when a data change causes an XPath expression to be evaluated.

For example, consider an XPath expression that contains the function **`mt-audio-is-playing`**. This function returns either **`true`** or **`false`**. When the expression is evaluated at some given time, let us say that the return value is **`true`** (because audio is playing). If this value is displayed in the solution, then the value will not automatically change when the audio stops playing. For this to happen, the function will have to be called again so that the new value updates the value in the display.

The number of ways in which such values can be updated depends on the particular design mechanisms that have been used. One possible way to update such values is via the timer of the OnPageRefresh[390] event in conjunction with the Update Display[790] action.

**Note:** For a description of functions in Altova's general XPath extension function library, see the section Altova Extension Functions[1689]. (The general extension functions work with all Altova products, including MobileTogether.)

▼ mt-audio-get-current-position

```
mt-audio-get-current-position(ChannelNumber as xs:integer) as xs:decimal
```

Takes as its argument the channel number on which the target audio file is playing. It returns a decimal number that is the current audio playback position in seconds. Note that information about current position is available only after playback has started, so the function should be used only subsequent to playback-start.

*Usage*

```
mt-audio-get-current-position(2)
```

▼ mt-audio-get-duration

```
mt-audio-get-duration(ChannelNumber as xs:integer) as xs:decimal
```

Takes as its argument the channel number on which the target audio file is playing. It returns a decimal number that is the duration, in seconds, of that audio file. Note that information about duration is available only after playback has started, so the function should be used only subsequent to playback-start.

*Usage*

```
mt-audio-get-duration(5)
```

▼ mt-audio-is-playing

`mt-audio-is-playing(ChannelNumber as xs:integer) as xs:boolean`

Takes as its argument the channel number to test. It returns `true()`if an audio file is playing on that channel, `false()` otherwise.

*Usage*

```
mt-audio-is-playing(3)
```

▼ mt-audio-is-recording

*Description*

Returns `true()`if the client device is recording audio, `false()` otherwise.

*Usage*

```
mt-audio-is-recording()
```

▼ mt-available-db-connection-names

`mt-available-db-connection-names(FromSolution as xs:boolean) as item()*`

Returns the names of all available database (DB) connections. If `FromSolution` is set to `true()`, returns the names of DB connections in the solution. If `FromSolution` is set to `false()`, returns the names of DB connections saved on the server. The returned object is a sequence of strings.

⊟ *Examples*

- `mt-available-db-connection-names( true() )` returns, for example, (**"MyCars", "companySales", "companyContacts")**
- `mt-available-db-connection-names( false() )` returns, for example, (**"DBConnOnServer-1", "DBConnOnServer-2")**

▼ mt-available-languages

*Description*

Returns the languages that have been defined in the Localization dialog 1600.

In the Localization dialog, each language is specified by its ISO language code (for example: **en-US**) and a given name (for example: **English**). The language code of the default language is always the empty string, but the name can be any given string.

The function returns each language as an array of two strings; for example: `[ "en-US", "English" ]`. Multiple languages are returned as a sequence of array items; for example: `( ["en-US", "English"], ["de-DE", "German (DE)"] )`. The first array in the sequence will always return the default language of the design. So if no language name is specified for the default language (in the Localization dialog), then both strings of the first array item will be empty; otherwise the first array item will contain an empty string and the name that was given for the default language (for example: `[ "", "MyDefaultLanguage" ]`).

Note that the returned sequence will be displayed as strings separated by spaces.

*Usage*

`mt-available-languages()` might return a display value of: `en-US English`
`mt-available-languages()` might return a display value of: `en-US English de-DE German (DE)`
`mt-available-languages()` might return a display value of: `MyDefaultLanguage en-US English`

▼ mt-base64-to-hexBinary

**mt-base64-to-hexBinary(Base64Image** *as xs:base64Binary*) **as xs:string**
The function converts a Base64-encoded image to a hexBinary string. The `Base64Image` argument must be text that is encoded in `base64Binary`. A page source node that provides such text can be submitted.

*Usage*
`mt-base64-to-hexBinary($XML1/Element1/@image)` converts a Base64 image to hexBinay

### mt-bluetooth-started

*Description*
Returns `true()` if Bluetooth has been started on the device and the solution has Bluetooth access, `false()` if the solution has no Bluetooth access.

*Usage*
`mt-bluetooth-started()`

### mt-cache-update-dateTime

*Description*
Returns the time when the page-source cache was updated. If the page source is not cached then an empty sequence is returned.

*Usage*
`mt-cache-update-dateTime($XML1)`

### mt-called-by-enter-key

**mt-called-by-enter-key() as xs:boolean**
Returns **`true()`** if the current action stack was triggered by pressing the **Enter** key, **`false()`** otherwise. Knowing that the user pressed the **Enter** key to start the current action stack—as opposed to, say, the **Esc** key—would indicate user intention, which in turn would enable conditional processing within the action stack.

### mt-called-by-escape-key

**mt-called-by-escape-key() as xs:boolean**
Returns **`true()`** if the current action stack was triggered by the user pressing the **Esc** key, **`false()`** otherwise. Knowing that the user pressed the **Esc** key would indicate different user intention than when the **Enter** key were pressed, thus enabling conditional processing within the action stack.

### mt-change-image-colors

**mt-change-image-colors(Base64Image** *as xs:base64Binary*, **SourceColors** *as xs:string+*, **TargetColors** *as xs:string+*, **Quality** *as xs:integer*) **as xs:base64Binary**
The function takes a Base64-encoded image as its first argument, changes those image colors that are submitted as the `SourceColors` argument into the corresponding `TargetColors`, and returns the transformed image as a Base64-encoded image.

- `Base64Image` must be text encoded in `base64Binary`. A node that returns such text can be submitted.
- `SourceColors` and `TargetColors` must be sequences with one or more string items. The number of items in both sequences must be the same.
- Quality is an integer from `1` to `100`. It specifies the quality level, with `100` being highest quality.

⊟ *Examples*

- `mt-change-image-colors(Base64ImageNode, ('#000000'), ('#666666'), 90 )` returns a Base64 image with black (`#000000`) turned to gray (`#666666`)
- `mt-change-image-colors(xs:base64Binary(Base64ImageNode), ('#000000', '#FF0000'), ('#666666', 'blue'), 90 )` returns a Base64 image with black (`#000000`) changed to gray (`#666666`) and red (`#FF0000`) changed to blue

▼ mt-client-ip-address

**`mt-client-ip-address() as xs:string`**

If an HTTP header, such as **`Remote_Addr`** or **`X-Forwarded-For`**, is used to make the HTTP request to MobileTogether Server, then **`mt-client-ip-address()`** returns the value of the header. Obtaining the value of the header is useful when the solution on a MobileTogether Server is behind a proxy or reverse proxy server.

If no relevant HTTP header is present, then the function returns the IP address of the client as seen from the server.

For simulations, set an IP address in the *Simulation* tab of the Options dialog (**Tools | Options**) of MobileTogether Designer.

⊟ *Example*

- `mt-client-ip-address()` returns the value of a relevant source header; otherwise, returns the client IP address. In simulations, returns the value set in the Options tab of MobileTogether Designer.

▼ mt-client-theme

**`mt-client-theme() as (xs:string, xs:string)`**

Returns a sequence of two strings. The first string is the theme that is currently used by the solution; this string can have a value of **`light`** or **`dark`**. The second string is the solution's theme setting as set in the project's properties 296; it can have a value of **`light`**, **`dark`**, or **`default`**.

⊟ *Example*

- `mt-client-theme()` returns **`("dark", "default")`**

▼ mt-connected-via-lan

*Description*

Returns `true()` if the mobile device is connected via LAN, `false()` otherwise.

*Usage*

`mt-connected-via-lan()`

▼ mt-connected-via-wifi

*Description*

Returns `true()` if the mobile device is connected via WiFi, `false()` otherwise.

*Usage*
`mt-connected-via-wifi()`

### mt-control-text-offset

**`mt-control-text-offset(ControlKind`** *`as xs:string`***`) as xs:integer*`**
**`mt-control-text-offset(ControlKind`** *`as xs:string,`* **`Parameters`** *`as map`***`) as xs:integer*`**

Returns the pixel value of the left, top, right, and bottom offsets, in that order, of the content of the control named in the **`ControlKind`** argument. The second argument, **`Parameters`**, is optional and is a ***`key-value`*** map that defines the properties of the control. The available keys and their values are listed below. The map filters the control kind named in **`ControlKind`** to those instances of the control kind that match the property values listed in the map.

The return values of this function can be used to align the content of the selected controls. See example below.

Given below are the available ***`key-value`*** pairs that can be submitted as the map of the **`Parameters`** argument. The order of the key-value pairs in the map is not fixed.

- `"Text Size" : "small"|"medium"|"large"`
- `"Unit" : "px"|"dp"|"sp"|""` *default is* `"px"`. For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.
- `"Bold Text" : "true"|"false"`
- `"Italic Text" : "true"|"false"`
- `"Underline Text" : "true"|"false"`
- `"Button Image" :` Any of the `Button Image` 417 options (for example, `"+"|"-"|">"|"Share"`)
- `"Button Background" : "transparent"|"non-transparent"`. Default is `"non-transparent"`.

*Example*

```
declare function AllOffsets($controlAsName)
  {
    let $map := map{
        'Label:': map{},
        'Edit Field:': map{},
        'Button:': map{},
        'Button:WithImage': map{'Button Image': 'Calendar'},
        'Button:WithImageTransparent': map{'Button Image': 'Calendar', 'Button
Background' : 'transparent'},
        'Button:WithImageRight': map{'Button Image': 'Calendar', 'Button Image
Position': 'right of text'},
        'Button:WithImageTransparentRight': map{'Button Image': 'Calendar', 'Button
Background' : 'transparent', 'Button Image Position': 'right of text'},
        'Combo Box:': map{},
        'Check Box:': map{},
        'Check Box:Right': map{'Check Mark Position': 'right of text'},
        'Radio Button:': map{},
        'Radio Button:Right': map{'Check Mark Position': 'right of text'},
        'Date:': map{},
        'Time:': map{},
        'Switch:': map{}
    }
    return element Root {
        for-each(map:keys($map), function($key) {
           let $offset := mt-control-text-offset(substring-before($key, ':'),
$map($key))
           let $name := if ($controlAsName) then 'control' else replace($key,
'[: ]', '')
           return element {$name} {
               attribute name {replace($key, ':', '')},
               attribute offsets {$offset},
               attribute left {$offset[1]},
               attribute top {$offset[2]},
               attribute right {$offset[3]},
               attribute bottom {$offset[4]}
           }
        })
    }
  }
```

### mt-control-width

**mt-control-width(Text** *as xs:string**, **Parameters** *as map(*)*) **as xs:integer?**

Returns the minimum width in pixels of the control when the **Text** string is the display-text of the control. The **Text** argument is the text that is displayed in the control. The **Parameters** argument is a ***key-value*** map that defines the properties of the control. The available keys and their values are listed below. The integer that is returned is the minimum width, in pixels, of the control when the submitted **Text** string is displayed with the properties specified in the **Parameters** argument. This value can then be used to calculate and specify other properties related to the control, such as the widths of table columns in which the control appears.

**Note:** This function is not available for web-client rendering. For web-client rendering, use the Measure Controls 927 action.

**Note:** This function can only be used in the XPath expressions of (i) design actions and (ii) the *Ensure Exists on Load (XPath Value)* option of page source tree nodes 371. It is not allowed in the XPath expressions of style properties.

Given below are the available ***key-value*** pairs that can be submitted as the map of the **Parameters** argument. The order of the key-value pairs in the map is not fixed. If a control property, as specified by a key-value pair, is not submitted, then that property's default value (respectively for Label 554 or Button 417 control) is used. Consequently, only the `Control Kind` parameter is mandatory.

- `"Control Kind" : "Label"|"Button"`
- `"Text Size" : "small"|"medium"|"large"`
- `"Unit" : "px"|"dp"|"sp"|""` *default is* `"px"`. For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP* 1312.
- `"Bold Text" : "true"|"false"`
- `"Italic Text" : "true"|"false"`
- `"Underline Text" : "true"|"false"`
- `"Button Image" :` Any of the `Button Image` 417 options (for example, `"+"|"-"|">"|"Share"`)
- `"Button Background" : "transparent"|"non-transparent"`. Default is `"non-transparent"`.

⊟ *Examples*

- **mt-control-width**("Send", map{"Control Kind" : "Button", "Text Size" : "medium", "Unit" : "", "Bold Text" : true(), "Italic Text" : false(), "Underline Text" : false(), "Button Image" : "+", "Button Background" : "transparent"})

### mt-convert-units

**mt-convert-units(Size** *as xs:string*, **TargetUnit** *as xs:string*) **as xs:string**

Converts the length value specified in the **Size** argument to an equivalent value in the unit specified by the **TargetUnit** argument. You can convert between any two of the following units: **px**, **dp**, and **sp**. Both input arguments, as well as the output value, are strings. For more information about units and conversion between them, see Sizes: Pixels, DPI, DP, SP 1312.

⊟ *Examples*

- **mt-control-unit("24px", "dp")** returns, depending on the device's resolution, for example, **"22dp"** on one device and **"20dp"** on another
- **mt-control-unit("20sp", "px")** returns, depending on the device's resolution, for example, **"22px"** on one device and **"24px"** on another

### mt-db-any-changed-fields

*Description*

Returns `true` if the row element contains new, modified or deleted columns. Returns `false` if the fields are unmodified. The function tests whether there was any modification to the specified DB row. *Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*

```
mt-db-any-changed-fields($DB1/DB/RowSet/Row[3])
```

### mt-db-any-changed-rows

*Description*

Returns `true` if the `$DB` variable (which represents a DB source) has new, modified or deleted rows. Returns `false` if the DB has not been modified. The function tests whether there was any modification to the DB, and works also if the `RowSets` element does not have a primary key. *Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*

```
mt-db-any-changed-rows($DB1)
```

### mt-db-deleted-original-fields

*Description*

Takes a **RowSet** row as its input and returns field attributes from the original `Row` element:

- *For new rows:* the function returns no field attributes. If the function is called for a new row, it returns an empty list.
- *For modified rows:* the function returns deleted-field attributes. If the function is called for a modified row, it returns those fields from the corresponding original row that are not listed in the `RowSet` element.

- *For deleted original rows:* the function returns all field attributes. If the function is called for a row that is deleted in the **`RowSet`**, it returns all fields of the deleted original row.

*Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*

`mt-db-deleted-original-fields($DB1/DB/RowSet/Row[1])`

### mt-db-deleted-original-rows

*Description*

Returns all `OriginalRow` elements for which no `Row` element exists. The function can be used to determine the modifications to data that was read from the database. *Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*

`mt-db-deleted-original-rows($DB1)`

### mt-db-file-path

*Description*

Takes a page source (variable) as its single parameter and returns the file path of the page source's file-based database. The file path will be relative to the Solutions Working Directory.

*Note*

- This is s server-side function and can be executed on the server only.
- The function works for SQLite and Access databases.

*Usage*

`mt-db-file-path($DB1)`

### mt-db-modified-fields

*Description*

Takes a **`RowSet`** row as its input and returns modified field attributes of the specified `Row` element:

- *For new rows:* all field attributes. If the function is called for a new row, it returns all fields.
- *For deleted original rows:* all field attributes. If the function is called for an `OriginalRow` (one for which the corresponding `Row` element was deleted), it returns all fields.
- *For modified rows:* the modified field attributes. If the function is called for a modified row, it returns those fields that have a different value from the one stored in the corresponding `OriginalRow` element.

*Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*

`mt-db-modified-fields($DB1/DB/RowSet/Row[3])`

### mt-db-modified-rows

*Description*

Takes a DB source as its input and returns a sequence of modified `Row` elements, that is, every `Row` element that is different under the `RowSet` element than the corresponding row in the `OriginalRowSet` element. The function can be used to determine modifications to current row data compared to original row data that was read from the DB. *Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

Note that rows, for this function, are identified by their primary keys. This function only checks whether content of the row has changed. To check for new rows, use the **`mt-db-new-rows`** function. To check for deleted rows, use the **`mt-db-deleted-original-rows`** function. A re-ordering of rows is not considered to be a modification.

*Usage*

```
mt-db-modified-rows($DB1)
```

### ▼ mt-db-new-fields

*Description*

Takes a **`RowSet`** row as its input and returns new field attributes of the specified `Row` element:

- *For new rows:* all field attributes. If the function is called for a new row, it returns all fields.
- *For modified rows:* the new field attributes. If the function is called for a modified row, it returns those fields that are not listed for the corresponding `OriginalRow` element.
- *For original rows:* an empty list. If the function is called for an `OriginalRow` element (one for which the corresponding `Row` element was deleted), it returns an empty list.

*Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*

```
mt-db-new-fields($DB1/DB/RowSet/Row[1])
```

### ▼ mt-db-new-rows

*Description*

Takes a DB source as its input and returns a list of new `Row` elements, that is, the `Row` elements that are listed under the `RowSet` element but not under the `OriginalRowSet` element. The function can be used to determine modifications to data that was read from the DB. *Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*

```
mt-db-new-rows($DB1)
```

### ▼ mt-db-original-row

*Description*

In order to enable the editing of DB rows, a copy of the original row set is saved in the DB page source in an element named **`OriginalRowSet`**, while modifications are stored in a parallel data structure named **`RowSet`**. On saving the page source, the content of **`RowSet`** is copied to **`OriginalRowSet`**.

The **`mt-db-original-row`** function takes a row from the **`RowSet`** data structure and returns the corresponding row from the original tree. This function is the opposite of the **`mt-db-row-from-original`** function. For more information, see the sections Editing DB Data 1031 and Saving Data to the DB 1035.

*Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*
`mt-db-original-row($DB1/DB/RowSet/Row[1])` returns `$DB1/DB/OriginalRowSet/Row[1]`.

▼ mt-db-row-from-original

*Description*
In order to enable the editing of DB rows, a copy of the original row set is saved in the DB page source in an element named **`OriginalRowSet`**, while modifications are stored in a parallel data structure named **`RowSet`**. On saving the page source, the content of **`RowSet`** is copied to **`OriginalRowSet`**.

The **`mt-db-row-from-original`** function takes a row from the original tree and returns the corresponding row in the **`RowSet`** data structure. This function is the opposite of the **`mt-db-original-row`** function. For more information, see the sections Editing DB Data 1031 and Saving Data to the DB 1035.

*Note:* For this function to work correctly, the `OriginalRowSet` 957 element must be enabled on the DB page source.

*Usage*
`mt-db-row-from-original($DB1/DB/OriginalRowSet/Row[1])` returns `$DB1/DB/RowSet/Row[1]`.

▼ mt-email-attachment

**`mt-email-attachment(Filename`** *`as xs:string`*, **`Content`** *`as item()`*, **`ContentType`** *`as xs:string`***`)`** **`as array(*)`**
Prepares the XML, Base64, or text content that is provided by the **`Content`** argument as an email attachment. Whether the content is parsed as XML or Base64 or text is determined by the **`ContentType`** argument, which takes either `XML`, `Base64`, or `text` as its value. The filename that is associated with the attachment is given by the **`Filename`** argument.

**Note:** The `mt-email-attachment` is a requirement when using the *Dynamic Attachments* option of the Send Email To 693 and Share 699 actions.

**Note:** For emails that are sent as HTML, the email body must be correct HTML, that is, it must start with the `html` element. A valid body could be created for example with the following XPath/XQuery construct:
**`element html { element body { "Test" } }`**

**Note:** Attachments work with Android and iOS clients only.
⊟ *Examples*

- **`mt-email-attachment`**`('MTNewFeatures.txt', $XML2/Releases/Release[@date='2015-04-15']/Features, 'XML')` returns `the Features node`
- **`mt-email-attachment`**`('MTLogo.jpg', $XML4/Images/Image[@name='MTLogo'], 'Base64')` returns `an image file`

▼ mt-external-error-code

*Description*
Returns the error code of the last DB, Load, or Save action. Returns the native error code of the OS or database, such as `404` when a web page cannot be found.

*Usage*
`mt-external-error-code()`

▼ mt-external-error-text

*Description*
Returns the error text of the last DB action, or Load or Save action. The error text is the text that is provided along with the returned error code.

*Usage*
`mt-external-error-text()`

▼ mt-extract-file-extension

**`mt-extract-file-extension(FilePath`** *`as xs:string`*) **`as xs:string?`**
Returns the file extension (for example, `xml`) of the file in the file path submitted as the **`FilePath`** argument. The string submitted in the **`FilePath`** argument must have the lexical pattern of an absolute or relative file path. Note that you can use the **`mt-last-file-path`** function as the **`FilePath`** argument.

⊟ *Examples*

- `mt-extract-file-extension(/storage/emulated/0/Download/MyFile.xml)` returns `'xml'`
- `mt-extract-file-extension(mt-last-file-path())` returns the extension of the file name in the file path returned by the `mt-last-file-path()` function

▼ mt-extract-file-name

**`mt-extract-file-name(FilePath`** *`as xs:string`*) **`as xs:string?`**
Returns the file name (the part before the file type extension) of the file in the file path submitted as the **`FilePath`** argument. The string submitted in the **`FilePath`** argument must have the lexical pattern of an absolute or relative file path. Note that you can use the **`mt-last-file-path`** function as the **`FilePath`** argument.

⊟ *Examples*

- `mt-extract-file-name(/storage/emulated/0/Download/MyFile.xml)` returns `'MyFile'`
- `mt-extract-file-extension(mt-last-file-path())` returns the name part of the file name in the file path returned by the `mt-last-file-path()` function; *see previous example*

▼ mt-font-height

**`mt-font-height(TextSize`** *`as xs:string*`*, **`Unit`** *`as xs:string`*) **`as xs:string?`**
Returns the height in pixels of the size-in-words that is submitted as the **`TextSize`** argument. Allowed values for the **`TextSize`** argument are: `smallest|small|medium|large|largest`. The optional **`Unit`** argument specifies the units in which the returned numeric height is required; currently only pixel heights are returned.

Each platform/device has its own pixel-height for each size-in-words. The `mt-font-height` function therefore enables you to obtain the numeric values that correspond to each size-in-words of the device, and to then calculate another numeric value. For example, to get a size that is 120% larger than the numeric size that corresponds to **`'largest'`** on a device, use the following XPath expression: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the **`'largest'`** size. This value is then multiplied by **`1.2`** to obtain the numeric value that is 120% of the value that corresponds to **`'largest'`**.

**Note:** This function can only be used in the XPath expressions of (i) design actions and (ii) the *Ensure Exists on Load (XPath Value)* option of page source tree nodes 371. It is not allowed in the XPath expressions of style properties.

⊟ *Examples*

- `mt-font-height("small", "px")` returns `33` (the value will vary from client to client)
- `mt-font-height("smallest", "")` returns `27` (the value will vary from client to client)

▼ mt-format-number

**`mt-format-number`**`(`**`Number`** *`as xs:numeric`*`, `**`PictureString`** *`as xs:string`*`)` **`as xs:string`**

Takes a number as the first argument, formats it according to the second (`PictureString`) argument, and returns the formatted number as a string. This is useful for formatting difficult-to-read numbers into a format that is more reader-friendly. The picture string can also contain characters, such as currency symbols, and so can also be used to insert characters in the formatted output. If you wish to insert a zero at a digit position when no digit exists in the input number at that position, then use a zero in that digit position of the picture string (*see examples below*). If you do not wish to force a zero (or other character), use the hash symbol (#).

Digits before the decimal separator are never foreshortened. The decimal part of a number (to the right of the decimal separator) as well as the units digit (first digit to the left of the decimal separator) are rounded off if the picture string of the decimal part is shorter than the number of decimal places in the input number.

**Note:** The grouping separator and decimal separator in the formatted output on the mobile device will be those of the language being used on the mobile device.

⊟ *Examples*

- `mt-format-number(12.3, '$#0.00')` returns `$12.30`
- `mt-format-number(12.3, '$00.00')` returns `$12.30`
- `mt-format-number(12.3, '$0,000.00')` returns `$0,012.30`
- `mt-format-number(12.3, '$#,000.00')` returns `$012.30`
- `mt-format-number(1234.5, '$#,##0.00')` returns `$1,234.50`
- `mt-format-number(1234.5, '$#0.00')` returns `$1234.50`
- `mt-format-number(123.4, '$0')` returns `$123`
- `mt-format-number(1234.5, '$0')` returns `$1235`
- `mt-format-number(1234.54, '$0.0')` returns `$1234.5`
- `mt-format-number(1234.55, '$0.0')` returns `$1234.6`

▼ mt-geo-map-marker

**`mt-geo-map-marker`**`(`**`id`** *`as xs:string`*`, `**`geolocation`** *`as xs:string`*`)` **`as map (*)`**
**`mt-geo-map-marker`**`(`**`id`** *`as xs:string`*`, `**`geolocation`** *`as xs:string`*`, `**`popup?`** *`as (xs:string*)`*`)` **`as map (*)`**
**`mt-geo-map-marker`**`(`**`id`** *`as xs:string`*`, `**`geolocation`** *`as xs:string`*`, `**`popup?`** *`as (xs:string*)`*`, `**`color?`** *`as xs:string`*`)` **`as map (*)`**

The function generates an XPath map-construct*****, which is used to create a marker for the Geolocation Map control. The function's **`id`** and **`geolocation`** arguments are mandatory; the **`popup`** and **`color`** arguments are optional. Each of the submitted strings is returned as text in a value of one of the map-construct's key–value pairs *(see the examples below)*. Each key of the returned map-construct receives its value from the corresponding argument of the function, these correspondences being determined by the index number of the argument. For example: the first argument of the function provides the value of the **`id`** key. Note that the function's third argument is a sequence of strings. These strings are used to generate the title and text of the marker's popup. The first string provides the popup's title; the following strings are concatenated to provide the text of the popup, with each string starting a new line. If you do not want a popup to be created, submit an empty sequence as the third argument. The **`color`** argument can be given as text (for example, **`"green"`**) or as an RGB value (for example, **`"#336699"`**). If no **`color`** argument is specified, then the default marker color of the device will be used.

Note that each function returns one marker. To generate multiple markers, use a sequence of multiple **`mt-geo-map-marker`** functions. *See examples below.* Note also that you can use the **`mt-geo-map-marker`** function within other XPath expressions as shown in the third example below (which uses an **`if-then-else`** construction).

***** *XPath map-construct: An XPath datatype construction similar to an XPath array. The map-construct is a sequence of key–value pairs (see the example map that is returned in the first example of the Usage section below). Note that the semantics of "map" in "XPath map-construct" refers to a mapping of keys to values in XPath; it does not refer to a cartographic map.*

<u>*Usage*</u>

**`mt-geo-map-marker("vie","48.2143531 16.3707266", ("Vienna","Altova EU","European headquarters"), "green")`** returns a single XPath map-construct—which, in turn, generates a single marker in the Geolocation Map control:

```
map {
     "id":"vie",
     "geolocation":(48.2143531, 16.3707266),
     "title":"Vienna",
     "text":"Altova EU
             European headquarters",
     "color":"green"
}
```

```
mt-geo-map-marker("vie","48.2143531 16.3707266", ("Vienna","Altova EU")), mt-geo-map-
marker("bev","42.5584577 -70.8893334", ("Beverly","Altova US"))
```

returns two XPath map-constructs—and so two markers for the Geolocation Map control.

```
mt-geo-map-marker("vie","48.2143531 16.3707266", ("Vienna","Altova EU")),
mt-geo-map-marker("bev","42.5584577 -70.8893334",("Beverly","Altova US")),
if ( $XML/MapMarkers/@withLondon = "1" ) then
     mt-geo-map-marker("lon","51.50939 -0.11832", ("London","No Altova") )
else
   ()
```

returns two XPath map-constructs, plus—if the **`@withLondon`** attribute has a value of **`"1"`**—a third XPath map-construct for the London geolocation. The function in effect generates two, or three, markers for the Geolocation Map control.

▼ mt-geolocation-started

<u>*Description*</u>

Returns `true()` if the solution has started geolocation tracking[735], `false()` otherwise.

*Usage*
`mt-geolocation-started()`

### mt-get-page-source-from-name

`mt-get-page-source-from-name(PageSourceName as string) as node`
Returns the root element of the page source having the name that matches the submitted string. Consequently, you can access nodes of the entire document tree via XPath. If no page source on the current page has a name that matches the submitted string, then an error is thrown. The submitted string argument does not need to include the leading `$` character.

*Usage*
`mt-get-page-source-from-name("$PERSISTENT")` returns the root element of the `$PERSISTENT` page source, for example `$PERSISTENT/Orders`
`mt-get-page-source-from-name("PERSISTENT")` returns the root element of the `$PERSISTENT` page source, for example `$PERSISTENT/MyRootElement`

### mt-get-page-source-name

`mt-get-page-source-name(PageSource as node()) as xs:string`
Returns the name of the page source that is submitted with the `PageSource` argument. The name is returned as a string.

*Usage*
`mt-get-page-source-name($XML1)` returns the string `"$XML1"`
`mt-get-page-source-name($PERSISTENT)` returns the string `"$PERSISTENT"`

### mt-get-page-source-structure

`mt-get-page-source-structure(PageSource as node() or as xs:string) as xs:string`
`mt-get-page-source-structure(PageSource as node() or as xs:string, EnsureValues as boolean) as xs:string`
`mt-get-page-source-structure(PageSource as node() or as xs:string, EnsureValues as boolean, Path as xs:string) as xs:string`

Returns the structure of the page source that is submitted with the `PageSource` argument. If the `EnsureValues` argument is set to `true()`, then the content of nodes is also returned. If `EnsureValues` is set to `false()`, or if no second argument is supplied (*see first signature above*), then the page source structure is returned without content. The structure and content are those at the time the page source is loaded.

A third argument (`Path`) can be specified as a string if you want to select a particular substructure of the tree returned by the first parameter, `PageSource`. The string will be evaluated as an XPath locator expression within the context of the node returned by `PageSource`. If the node returned by this expression is an element, then the element will be returned with its attributes but without any child element.

*Usage*
`mt-get-page-source-structure($XML1)` returns the data structure of the `$XML1` page source when the page source was loaded, but without values
`mt-get-page-source-structure($XML1, true())` returns the data structure of the `$XML1` page source, with values
`mt-get-page-source-structure($XML1/Companies/Company, false(), "Departments/Department)`

returns the `Department` element with its attributes, but without values.
`for $i in mt-get-page-source-structure($XML1, true()) return $i//Product[1]` returns the content of the first `Product` element of the `$XML1` page source at the time when the page source was loaded

▼ mt-has-serveraccess

*Description*
Returns `true` if server access is possible, `false` otherwise. There are two signatures:

- `mt-has-serveraccess(`**`TimeoutSeconds`** as `integer)`: The function checks whether a connection to MobileTogether Server can be established within the number of seconds specified by the `TimeoutSeconds` argument of the function.
- `mt-has-serveraccess(`**`URL`** as `string,` **`TimeoutSeconds`** as `integer)`: Checks whether a connection to the server located at the **`URL`** argument can be established within the number of seconds specified by the `TimeoutSeconds` argument of the function. Essentially, a **`GET`** request to the server URL is performed with a timeout.

*Usage*

```
mt-has-serveraccess(5)
mt-has-serveraccess('https://www.altova.com', 5)
```

▼ mt-hexBinary-to-base64

**`mt-hexBinary-to-base64(HexBinary`** *`as xs:string`***`) as xs:base64Binary`**
The function converts a hexBinary string to a Base64-encoded string (typically an image). A node that provides the required hexBinary string can be submitted as the `HexBinary` argument.

*Usage*
`mt-hexBinary-to-base64('48656C6C6F20576F726C64')` returns the Base64 string `'SGVsbG8gV29ybGQ='`

▼ mt-hexBinary-to-string

**`mt-hexBinary-to-string(HexBinary`** *`as xs:string,`* **`Encoding`** *`as xs:string`***`) as xs:string`**
**`mt-hexBinary-to-string(HexBinary`** *`as xs:string`***`) as xs:string`**
The function converts a hexBinary string to a text string that is encoded with the encoding named in the **`Encoding`** argument. A node that provides a hexBinary string can be submitted as the **`HexBinary`** argument. If the empty string is supplied as the `Encoding` argument or if no `Encoding` argument is specified, then the result text string is generated in the default **`'UTF-8'`** encoding.

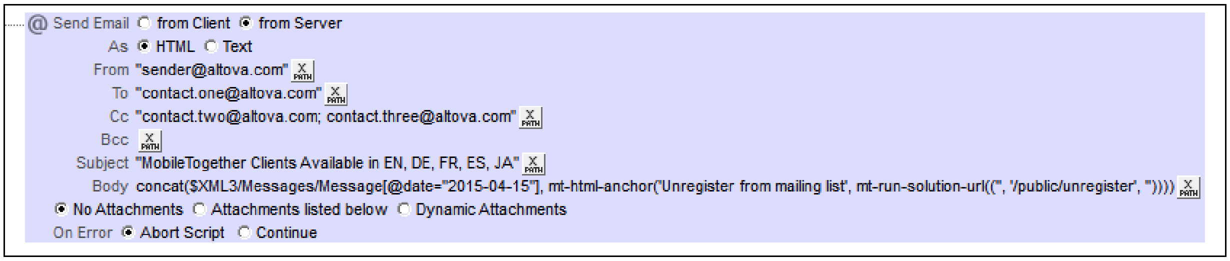
⊟ *Examples*

- `mt-hexBinary-to-string('48656C6C6F20576F726C64', 'ASCII')` returns `'Hello World'`
- `mt-hexBinary-to-string('48656C6C6F20576F726C64', '')` returns `'Hello World'`
- `mt-hexBinary-to-string('48656C6C6F20576F726C64')` returns `'Hello World'`

▼ mt-html-anchor

*Description*
The `mt-html-anchor` 1262 function takes two arguments: `LinkText` and `TargetURL`. It uses these two arguments to create an HTML hyperlink element: `<a href="TargetURL">LinkText</a>`. The link can be

inserted in emails that are sent as HTML by using the Send Email To[693] action. The link can open an Internet page or a MobileTogether solution. To add a link to the email body, use the `mt-html-anchor`[1262] function in the XPath expression of the *Body* option (*see screenshot below*).



@ Send Email ○ from Client ● from Server
As ● HTML ○ Text
From "sender@altova.com"
To "contact.one@altova.com"
Cc "contact.two@altova.com; contact.three@altova.com"
Bcc
Subject "MobileTogether Clients Available in EN, DE, FR, ES, JA"
Body concat($XML3/Messages/Message[@date="2015-04-15"], mt-html-anchor('Unregister from mailing list', mt-run-solution-url('', '/public/unregister', '')))
● No Attachments ○ Attachments listed below ○ Dynamic Attachments
On Error ● Abort Script ○ Continue

⊟ *Examples*

- `mt-html-anchor('Unregister from mailing list', 'http://www.altova.com/unregister.html')` returns `<a href="http://www.altova.com/unregister.html">Unregister from mailing list</a>`
- `mt-html-anchor('Unregister from mailing list', mt-run-solution-url('', '/public/unregister', ''))` returns `<a href="LinkTo-unregister.mtd">Unregister from mailing list</a>`

▼ mt-image-width-and-height

**`mt-image-width-and-height(Image`** *`as base64encoded-image`* **`) as xs:integer+`**
The argument `Image` is the Base64-encoding of the image, the dimensions of which you wish to know. The argument must be of type `xs:base64Binary`. Typically, the argument would locate a node that contains the Base64-encoded data. The function returns a sequence two integers: (i) the width of the image in pixels, (ii) the height of the image in pixels.

⊟ *Examples*

- `mt-image-width-and-height($XML1/images/png)` might return `364 76`
- `mt-image-width-and-height($XML1/images/png) > 500` returns `true()` if at least one value (width or height) is greater than 500
- `mt-image-width-and-height($XML1/images/png) > 500` returns `false()` if both values (width and height) are not greater than 500

▼ mt-in-app-purchase-platform-to-product

**`mt-in-app-purchase-platform-to-product(PlatformID`** *`as xs:string`* **`) as xs:string`**
The argument `PlatformID` is the ID of a product on a given platform. It returns the product name that maps to this product ID, as defined in the In-App Purchase Products dialog[1505]. Also see the function **`mt-in-app-purchase-product-to-platform()`**.

⊟ *Examples*

- `mt-in-app-purchase-platform-to-product('AndroidID_For_MyBlogSubscription')` returns, on Android devices: `'MyBlog-Subscription'`
- `mt-in-app-purchase-platform-to-product('AppleID_For_MyBlogSubscription')` returns, on iOS devices: `'MyBlog-Subscription'`
- `mt-in-app-purchase-platform-to-product('WindowsID_For_MyBlogSubscription')` returns,

on Windows devices: `'MyBlog-Subscription'`

▼ mt-in-app-purchase-product-to-platform

**`mt-in-app-purchase-product-to-platform(ProductName`** *`as xs:string`* **`) as xs:string`**

The argument `ProductName` is the name of a product as defined in the In-App Purchase Products dialog 1505. It returns the ID of that product on the current platform according to the mapping in the In-App Purchase Products dialog 1505. Also see the function **`mt-in-app-purchase-platform-to-product()`**.

⊟ *Examples*

- `mt-in-app-purchase-product-to-platform('MyBlog-Subscription')` returns, on Android devices: `'AndroidID_For_MyBlogSubscription'`
- `mt-in-app-purchase-product-to-platform('MyBlog-Subscription')` returns, on iOS devices: `'AppleID_For_MyBlogSubscription'`
- `mt-in-app-purchase-product-to-platform('MyBlog-Subscription')` returns, on Windows devices: `'WindowsID_For_MyBlogSubscription'`

▼ mt-in-app-purchase-service-started

**`mt-in-app-purchase-service-started() as xs:boolean`**

Checks whether the client device has all the necessary requirements to carry out in-app purchases at its respective app store, including a running billing service. For example, on Android, a user account for the app store is required to carry out an in-app purchase. The function returns **`true()`** if the service is available, **`false()`** if not.

⊟ *Examples*

- `mt-in-app-purchase-service-started()` returns `true()` if all requirements for in-app purchases of the relevant app store are met, `false()` if they are not

▼ mt-invert-color

**`mt-invert-color(Color`** *`as xs:string`* **`) as xs:string`**

The argument `Color` is the RGB color code (in hexadecimal format). For example `"#00FFFF"`. Each color component (R, G, and B) in the code is inverted, and the new color code is returned.

⊟ *Examples*

- `mt-invert-color('#000000')` returns `'#FFFFFF'`
- `mt-invert-color('#00FFFF')` returns `'#FF0000'`
- `mt-invert-color('#AA0000')` returns `'#55FFFF'`
- `mt-invert-color('#AA33BB')` returns `'#55CC44'`
- `mt-invert-color('#34A6D2')` returns `'#CB592D'`

▼ mt-is-server-purchased

*Description*

Returns `true` if all the licenses assigned to MobileTogether Server are purchased licenses, `false` if one or more assigned licenses is a trial one. Tip: If a trial license is no longer needed, unassign it.

Note the following:

- On clients the function returns `false` by default. Only when a request is made to a server will the purchase-state of the server's licenses be returned.
- In the simulator, the function always returns `false`. Use the corresponding simulator option 1668 to simulate a purchased license. For simulations in the designer and for trial runs on client, switching this option on simulates that MobileTogether Server licenses have been purchased. For simulations on the server, the actual purchase-state of licenses on the server is returned.

*Usage*
`mt-is-server-purchased()`

### mt-last-file-path

**`mt-last-file-path()` `as xs:string?`**
Returns the full file path (for example, on Android: `/storage/emulated/0/Download/MyFile.xml`) of the last client file used (loaded or saved) in any of the following actions: Audio Recording 724, Video Recording 729, Load/Save File 809, Load/Save Binary File 815, Load Image 707. Note that some versions of operating systems other than Android might not support this function.

*Usage*
`mt-last-file-path()`

### mt-last-in-app-purchase-response-code

**`mt-last-in-app-purchase-response-code()`** as **`xs:integer`**
Returns the success/failure code of the last in-app purchase request made by the client device at the app store. If the request was successfully executed, the code would typically be `0`, otherwise it would return some other integer. The function is useful for checking whether a step of the workflow was executed correctly. Related functions are **`mt-in-app-purchase-response-text()`** and **`mt-in-app-purchase-response-was-user-canceled()`**.

⊟ *Examples*

- **`mt-in-app-purchase-response-code()`** typically returns `0` if the last in-app purchase request was correctly executed.

### mt-last-in-app-purchase-response-text

**`mt-last-in-app-purchase-response-text()` `as xs:string`**
Returns a text message describing the success/failure of the last in-app purchase request. The function is particularly useful if there is an error because it provides textual description of the error. Related functions are **`mt-in-app-purchase-response-was-user-canceled()`** and **`mt-in-app-purchase-response-code()`**.

⊟ *Examples*

- **`mt-in-app-purchase-response-text()`** returns a suitable text message according to the success level of the action at the app store

### mt-last-in-app-purchase-response-was-user-canceled

**`mt-last-in-app-purchase-response-was-user-canceled()` `as xs:boolean`**
Checks with the app store whether the last in-app purchase was canceled by the user. It returns **`true()`**if

the last purchase was canceled, **`false()`**otherwise. Related functions are **`mt-in-app-purchase-response-text()`**and **`mt-in-app-purchase-response-code()`**. This function is useful for distinguishing this kind of failed purchase request from other kinds of error that might occur during the purchase request.

⊟ *Examples*

- **`mt-in-app-purchase-response-was-user-canceled`**`()` returns `true()`if the user canceled the last in-app purchase, `false()` if they are not

### ▼ mt-load-json-from-string

*Description*

The function takes as its argument a string that is a serialized JSON structure. The JSON structure is converted to XML and wrapped inside an element named **`json`**, and returned as a document node. For example, say you submit the following JSON document in a string (that is, within quotes) as the argument of the function :

```
{
   "Root": {
      "user": "Altova",
      "message": "Hello",
      "A": {
         "B": "B Text",
         "C": "C Text",
         "D": {
         }
      }
   }
}
```

The function will return a document node having a document element named `json` that contains the JSON structure converted to XML. The child element of **`json`** will be **`Root`**.

Since it is a document node that is returned by the function, you can access parts of the returned document by appending an XPath locator expression to the function. For example, if you evaluate the following expression (with the locator expression highlighted in yellow) :

```
mt-load-json-from-string('{
      "A": {
         "B": "B Text",
         "C": "C Text",
         "D": {
         }
      }
}')/json/A/C
```

it returns **`"C Text"`**.

*Usage*

`mt-load-json-from-string('{ "A": "A Text" }')` returns the document node having a document element named **`json`**, which will have one child element named **`A`**

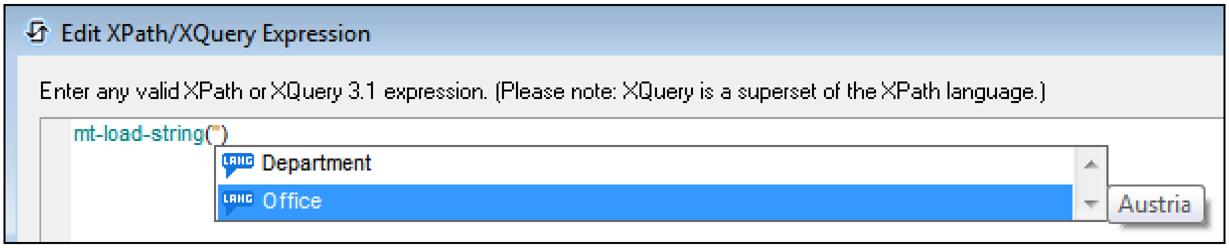
`mt-load-json-from-string('{ "A": "A Text" }')`**`/*`** returns all the elements of the document: that is, the **`json`** element with one child element named **`A`**.

mt-load-string

*Description*
Returns the custom string[1600] identified by the `StringName` argument. Each custom string[1600] is part of a pool of strings that are defined in the Localization dialog[1600]. Each `StringName` is associated, in the string pool, with multiple localized strings. The language of the localized string that is selected is the same as the language of the mobile device or simulation language[1606].

*Usage*
`mt-load-string('StringName')`

When the `mt-load-string` function is being entered in the Edit XPath/XQuery Expression dialog[1244], all available custom strings are displayed in a popup (*see screenshot below*). To display this popup, place the cursor inside the delimiting apostrophes or quotes of `'StringName'`, and click **Ctrl+Spacebar**.


Edit XPath/XQuery Expression
Enter any valid XPath or XQuery 3.1 expression. (Please note: XQuery is a superset of the XPath language.)
mt-load-string('')
Department
Office
Austria

Cycle through the list by using the **Up** and **Down** keyboard buttons. The value of the selected custom string[1600] is displayed to the right of the popup (*see screenshot above*). The localization language of the displayed value is that of the simulation language[1606] that is currently selected in MobileTogether Designer. To enter the name of a custom string[1600] in the XPath expression, select the string, or cycle through the custom string[1600] list to the string you want and press **Enter**.

mt-localized-string-name

**`mt-localized-string-name(Text as xs:string) as xs:string*`**
**`mt-localized-string-name(Text as xs:string, Lang as xs:string) as xs:string*`**
The  function takes as its (first) argument a text-string value in the default language or a localized language and returns the name of the control or string that has the submitted text-string value as its text value. See Localization[308] and **Project | Localization**[1600] for more information. The function has two signatures. In the second signature, the language of the text-string is the second argument (`Lang`). The `Lang` argument should match the name of a localized language. If `Lang` is specified, then the strings of that localized language only are searched for a text-string that matches the text-string submitted in the `Text` argument.

*Examples*

- `mt-localized-string-name('City')` returns `'CityButton'`
- `mt-localized-string-name('Stadt', 'DE')` returns `'CityButton'`
- `mt-localized-string-name('Stadt')` returns `'CityButton'`
- `mt-localized-string-name('Stadt', 'ES')` returns `''`
- `mt-localized-string-name('Stadt', 'German')` returns `''`
- `mt-localized-string-name('Ciudad', 'ES')` returns `'CityButton'`

The examples above are for a string on a Button control that is named **CityButton**. The default language of the string is English and it has been localized for languages named **DE** and **ES**.

### mt-nfc-started

*Description*
Returns true() if the solution has started NFC 746, false() otherwise.

*Usage*
mt-nfc-started()

### mt-page-stack

*Description*
Returns a list of open pages. The list will be a sequence of strings, that is, it will consist of the names of pages with a space between each pair of names.

*Usage*
mt-page-stack()

### mt-progress-cancelling

*Description*
This function is available only while progress indication actions 795 are being executed. The default is **false()**. It returns **true()** if the Progress Send Cancellation 797 action is triggered while a set of progress indication actions 795 is being executed. The value of this function can therefore be used to receive the client user's intention to cancel and consequently cancel server actions.

*Usage*
If mt-progress-cancelling()=true()then carry out action/s to stop server actions and take any other actions that such a cancellation might necessitate (for example, resetting a node value to a pre-action value).

### mt-refresh-userroles (deprecated)

*Description*
Loads the currently available user roles from the server. The function updates the user roles from the server that can be queried via the global variable MT_UserRoles 1304.

*Usage*
mt-refresh-userroles()

### mt-reload-dateTime

*Description*
Returns the time at which the page-source was reloaded. If not loaded then an empty sequence is returned.

*Usage*
mt-reload-dateTime($XML1)

### mt-run-appstoreapp-url

**mt-run-appstoreapp-url(Scheme?** *as xs:string*, **Host?** *as xs:string*, **InputParameters?** *as*

*xs:string*) **as xs:string?**
**mt-run-appstoreapp-url(InputParameters?** *as xs:string*) **as xs:string?**

Generates the URL of a MobileTogether AppStore App[1471] from either (i) the three submitted arguments, or (ii) the single `InputParameters` argument. On clicking the URL, which would typically be sent in an email, the AppStore App[1471] will be started. The URL must have the format: **`<url-scheme>://<url-host>`**. The app's manifest file contains the scheme information, and this indicates to the device that URLs starting with this scheme should be opened by this app. For more information about AppStore Apps[1471], see the section AppStore Apps[1471].

- `Scheme`: The unique scheme name that is associated with the app. The scheme is assigned when the app's program code is generated (in Screen 1 of the Generate Program Code Wizard[1472]). If this argument is omitted or an empty string is submitted, then the scheme is set to that of the currently running app.
- `Host`: The unique host name that is associated with the app. The host is assigned when the app's program code is generated (in Screen 1 of the Generate Program Code Wizard[1472]). If this argument is omitted or an empty string is submitted, then the hostname will be that associated with the currently running app.
- `InputParameters`: Takes the function `mt-run-solution-url-parameters` as its input. The argument of the function is a sequence of string values that provide the values of the query's parameters. The `mt-run-solution-url-parameters` function returns a string containing the parameters (names and values) of the URL's query string, correctly encoded and percent-escaped according to the rules for encoding URL query strings. The parameter names in the result string are automatically generated by the function (they are: `in1, in2 ... inN`), and each is assigned a value from the string items of the function's argument, with names and values being paired in index order. (Additionally, the `InputParameters` argument can be provided as a string that is already encoded for the query string part of a URL (*see second example below*).)

The `mt-run-appstoreapp-url` function therefore creates a URL, with or without query parameters, that opens a MobileTogether AppStore App[1471] . The query parameters are passed to the app when the app is opened via the URL. The values of these parameters can be accessed in other design components by using the **`$MT_InputParameters`**[1300] global variable.

⊟ *Examples*

- **mt-run-appstoreapp-url**('myappscheme', 'myfirstapp', '') returns the URL **`myappscheme://myfirstapp`**. On a mobile device, the URL will open the AppStore app that is identified by that scheme and host. The URL has no query parameters.
- **mt-run-appstoreapp-url**('myappscheme', 'myfirstapp', 'in1=value1&in2=value2%3FAndMoreValue2') returns a URL that opens AppStore app that is identified by that scheme and host. The `InputParameters` argument is submitted to the function as a string that is encoded as a URL query string.

▼ mt-run-solution-url

**mt-run-solution-url(ServerAddress?** *as xs:string*, **SolutionName?** *as xs:string*, **InputParameters?** *as xs:string*) **as xs:string?**

Generates a URL to open the specified solution in a MobileTogether client. When the URL is tapped, the Altova MobileTogether Client app is opened and the solution is started in the app. The URL is generated from either (i) the function's three submitted arguments *(listed below)*, or (ii) the function's `InputParameters` argument.

- `ServerAddress`: Takes the name or IP address of the MobileTogether Server on which the solution that you want to run is deployed. If this argument is omitted or it is the empty string, then the current server is used.
- `SolutionName`: Takes the deployed path of the solution on the server. For example: `/public/MySolution` (which would point to the `MySolution.mtd` file in the `/Public` container). If this argument is omitted or it is the empty string, then the current solution is used.
- `InputParameters`: Takes the function `mt-run-solution-url-parameters` as its input. The argument of the **`mt-run-solution-url-parameters`** function is either (i) a sequence of string values that will be the values of the query's parameters, or (ii) a map of *`key:value`* pairs that provide the name and value of the respective parameters. This function returns a string that is correctly encoded and percent-escaped according to the rules for encoding URL query strings. See the description of the **`mt-run-solution-url-parameters`** function below. (Additionally, the `InputParameters` argument can be provided as a string that is already encoded for the query string part of a URL (*see fourth example below*).)

The `mt-run-solution-url` function therefore creates a URL, with or without query parameters, that accesses a solution on a MobileTogether Server. The query parameters are passed to the solution when the solution is opened via the URL. The values of these parameters can be accessed in other design components by using the **`$MT_InputParameters`** 1300 global variable.

⊟ *Examples*

- **`mt-run-solution-url`**`('100.00.000.1', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the server with the IP address `100.00.000.1`. The URL has no query parameters.
- **`mt-run-solution-url`**`('', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the current server. The URL has no query parameters.
- **`mt-run-solution-url`**`('', '', mt-run-solution-url-parameters(('2015', 'USA', 'true')))` returns a URL that points to the current solution on the current server. The argument of the **`mt-run-solution-url-parameters`** function in this example is a sequence of string values that will be the values of the query's parameters. The first string will be the value of the first parameter, the second string will be the value of the second parameter, and so on. See the description of the **`mt-run-solution-url-parameters`** function below.
- **`mt-run-solution-url`**`('', '', 'in1=value1&in2=value2%3FAndMoreValue2')` returns a URL that points to the current solution on the current server. The `InputParameters` argument is submitted as a string already encoded as a URL query string.

Note the following points:

- The first argument, `ServerAddress`, is used to look up information on the client about a server having the submitted name/address. The port number, user name, and user password that are associated with the server name are then used to connect to the server. So if a URL is generated with a server name that is not recognized by the client, then the URL will not work.
- The second argument, `SolutionName`: (i) generates the deployed path (on the server) if the solution is run on the server, but (ii) generates a file path for simulations.
- The third argument, `InputParameters`, uses the MobileTogether-specific XPath extension function called **`mt-run-solution-url-parameters`** to generate and encode the query's parameter-value pairs. Do not confuse the **`mt-run-solution-url-parameters`** function (which encodes the query parameters) with the `mt-run-solution-url` function (which generates the whole URL).

### ▼ mt-run-solution-url-parameters

```
mt-run-solution-url-parameters((Parameters*) as xs:string) as xs:string?
mt-run-solution-url-parameters(Map as map) as xs:string?
```

The `mt-run-solution-url-parameters` function is intended for use as the third argument of the `mt-run-solution-url` function. Its single argument is either a sequence of string values or a map of key–value pairs. The result that is generated is a single string, which will be the query string part that is to be submitted (as an argument) to the `mt-run-solution-url` function. It contains the parameters (names and values) of the query string, correctly encoded and percent-escaped according to the rules for encoding URL query strings.

*Sequence*

The argument is a sequence of string values, which are the parameter values of the query string. The parameter names in the result string are automatically generated by the function (they are: `in1, in2 ... inN`), and each is assigned a value from the string items of the function's argument, with names and values being paired in index order. See the sequence examples below.

*Map*

The parameters (names and values) can also be submitted as a map of *`key:value`* pairs. For example: `map{"key1":"value1", "key2":"value2"}`. The ordering of parameters is not important because each parameter value is keyed to a specific parameter name. See the map examples below.

**Note:** If the `Parameters` string contains double quotes, change these to single quotes. This is required because MobileTogether uses double quotes to build the parameters string. You can use the XPath `replace` function to change double quotes to single quotes: **`replace(<string>, '"', "'")`**. Also see the *C'est la vie* examples below, where the text must be delimited by double quotes (not single quotes).

The values of these parameters can be accessed in other design components by using the **`$MT_InputParameters`** 1300 global variable.

⊟ *Example*

- `mt-run-solution-url-parameters(('2015', 'USA', 'true'))` returns `"in1=2015&in2=USA&in3=true"`
- `mt-run-solution-url-parameters(("2015", "USA", "true"))` returns `"in1=2015&in2=USA&in3=true"`
- `mt-run-solution-url-parameters(("2015", "", '', ""))` returns `"in1=2015&in2=&in3=&in4="`
- `mt-run-solution-url-parameters(("Stereophonics", "C'est la vie"))` returns `"in1=Stereophonics&in2=C#est%20la%20vie"`
- `mt-run-solution-url-parameters(map{ 'Year':'2015', 'Country:'USA', 'Include':'true') })` returns `"Country=USA&Include=true&Year=2015"`
- `mt-run-solution-url-parameters(map{ 'Artist':'Stereophonics', 'Title':"C'est la vie") })` returns `"Country=USA&Include=true&Year=2015"`

### ▼ mt-run-web-url

```
mt-run-web-url(ServerAddress? as xs:string, SolutionName? as xs:string,
InputParameters? as xs:string) as xs:string?
mt-run-web-url(InputParameters? as xs:string) as xs:string?
```

Generates a URL, which is entered in a web browser, to open the specified solution in the browser. The URL is generated from either (i) the function's three submitted arguments *(listed below)*, or (ii) the function's `InputParameters` argument:

- `ServerAddress`: Takes the name or IP address of the MobileTogether Server on which the solution that you want to run is deployed. If this argument is omitted or it is the empty string, then the current server is used.
- `SolutionName`: Takes the deployed path of the solution on the server. For example: `/public/MySolution` (which would point to the `MySolution.mtd` file in the `/Public` container). If this argument is omitted or it is the empty string, then the current solution is used.
- `InputParameters`: Takes the function `mt-run-solution-url-parameters` as its input. The argument of this function is a sequence of string values that provide the values of the query's parameters. The `mt-run-solution-url-parameters` function returns a string containing the parameters (names and values) of the URL's query string, correctly encoded and percent-escaped according to the rules for encoding URL query strings. The parameter names in the result string are automatically generated by the function (they are: `in1, in2 ... inN`), and each is assigned a value from the string items of the function's argument, with names and values being paired in index order. (Alternatively, the `InputParameters` argument can be provided as a string that is already encoded for the query string part of a URL (*see fourth example below*).)

The `mt-run-web-url` function therefore creates a URL, with or without query parameters, that accesses a solution on a MobileTogether Server. The query parameters are passed to the solution when the solution is opened via the URL. The values of these query parameters can be accessed in other design components by using the **`$MT_InputParameters`** 1300 global variable.

⊟ *Examples*

- **`mt-run-web-url`**`('100.00.000.1', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the server with the IP address `100.00.000.1`. The URL has no query parameters.
- **`mt-run-web-url`**`('', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the current server. The URL has no query parameters.
- **`mt-run-web-url`**`('', '', mt-run-solution-url-parameters(('2015', 'USA', 'true')))` returns a URL that points to the current solution on the current server. The argument of the **`mt-run-solution-url-parameters`** function is a sequence of string values that will be the values of the query's parameters. The first string will be the value of the first parameter, the second string will be the value of the second parameter, and so on. The **`mt-run-solution-url-parameters`** function returns a string that is correctly encoded and percent-escaped according to the rules for encoding URL query strings.
- **`mt-run-web-url`**`('', '', 'in1=value1&in2=value2%3FAndMoreValue2')` returns a URL that points to the current solution on the current server. The `InputParameters` argument is submitted as a string already encoded as a URL query string.

Note the following points:

- The first argument, `ServerAddress`, is used to look up information on the client about a server having the submitted name/address. The port number, user name, and user password that are associated with the server name are then used to connect to the server. So, if a URL is generated with a server name that is not recognized by the client, then the URL will not work.
- The second argument, `SolutionName`: (i) generates the deployed path (on the server) if the solution is run on the server, but (ii) generates a file path for simulations.
- The third argument, `InputParameters`, uses the MobileTogether-specific XPath extension function

called `mt-run-solution-url-parameters` to generate and encode the query's parameter-value pairs. Do not confuse the `mt-run-solution-url-parameters` function (which encodes the query parameters) with the `mt-run-solution-url` function (which generates the whole URL).

### mt-save-json-to-string

*Description*

Takes a JSON node as its single argument, and returns the contents of the node as a string that is serialized in JSON format. The submitted JSON node could be an entire JSON document, or a part of a JSON document: For example, say you have the following JSON document associated with a JSON page source named **`$JSON1`**:

```
{
   "Root": {
      "user": "Altova",
      "message": "Hello",
      "A": {
         "B": "B Text",
         "C": "C Text",
         "D": {
         }
      }
   }
}
```

If the document node is submitted (that is, **`$JSON1`**, which is the virtual node that is the document's parent node), then the contents of the entire document object is returned as a string. If the **`$JSON1/json//Root`** node is submitted as the argument, then the object that is the value of the **`"Root"`** key is returned as a string. Also see the examples below.

*Usage*

`mt-save-json-to-string($JSON/json/Root/A/B)` returns `'"B Text"'`
`mt-save-json-to-string($JSON/json/Root/A/D)` returns `'{}'`

### mt-server-config-url

**`mt-server-config-url(ServerSettings`** *`as map`* **`) as xs:string?`**

The **`mt-server-config-url`** function takes a map as its argument and returns a string that is a URL. When the URL is sent as a link to client devices, and the link is tapped, then server settings on the client are automatically updated. The URL will look something like this: `mobiletogether://mt/change-settings?settings=<json encoded settings>`

The JSON-encoded server settings that are contained in the URL are provided by the `ServerSettings` argument of the **`mt-server-config-url`** function. The `ServerSettings` map is shown below. For an example of how to use this function, open and test the example file **`ClientConfiguration.mtd`** in the `MobileTogetherExamples/SimpleApps` folder.

```
mt-server-config-url(
  map{
    "DelOthSrv": false(),   (: whether existing server list should be deleted before
import :)
    "DetView": true(),      (: whether the details view should be used or the grid :)
    "Refresh": true(),      (: refresh solutions on start :)
    "RetToSln": true(),     (: Windows clients only :)
    "ActSrvURL": "",        (: the first server with this URL gets the active one :)
    "Servers": array{
       map{
         "Name": "",
         "URL": "",         (: if DelOthSrv is false then this property is used as key
to merge the new settings with the existing ones :)
         "LoginProvider": map{
             "NameSuffix": "",
             "NamePrefix": "",
                             },
         "Port": "",
         "User": "",
         "StorePW": true(),
         "Password": "",
         "SSL": false()
          }                  (: , map {...} to add another server :)
                     }
     }
)
```

### mt-server-variable

**`mt-server-variable(VariableName as xs:string) as xs:string`**

Returns the value of the server variable named in the **`VariableName`** argument. Server variables are variables that are stored in the `mobiletogetherserver.cfg` file (*see the [MobileTogether Server user manual](#)*). In this file, the server variables are stored in the `[ServerVariables]` section, as shown in the snippet below:

```
[ServerVariables]
Environment=Portal
Manual=AdminDocs
StartPage=Admin
```

The `mobiletogetherserver.cfg` file is located by default in the MobileTogether Server application data folder. You can edit the `.cfg` configuration file in a text editor. See the [MobileTogether Server user manual](#) for more information.

For information about how to simulate server variables, see the Options tab [Simulation 2](#)[1669].

⊟ *Examples*

- `mt-server-variable("Environment")` returns, for example, `"Development"` or `"Portal"`

### mt-solution-path

**`mt-solution-path() as xs:string`**

Returns the path of the currently running solution.

### mt-string-to-hexBinary

**`mt-string-to-hexBinary(Text as xs:string, Encoding as xs:string) as xs:string`**

**mt-string-to-hexBinary(Text** *as xs:string***) as xs:string**
The function converts a text string to a hexBinary string. A node that returns a text string can be submitted as the **Text** argument. The function reads the **Text** string as having the encoding specified in the **Encoding** argument. If the empty string is supplied as the **Encoding** argument or if no **Encoding** argument is supplied, then the default encoding **'UTF-8'** is used.

⊟ *Examples*

- **mt-string-to-hexBinary**('Hello World', 'ASCII') returns '48656C6C6F20576F726C64'
- **mt-string-to-hexBinary**('Hello World', '') returns '48656C6C6F20576F726C64'
- **mt-string-to-hexBinary**('Hello World') returns '48656C6C6F20576F726C64'

## mt-table-rowgroup-count

**mt-table-rowgroup-count(VisibleOnly?** *as xs:boolean***) as xs:integer**
The function must be set inside a row-group, which becomes the context of the function. The function returns the number (count) of row-groups in the table within which the context row-group is located. The **VisibleOnly** argument is optional; it can have a value of true() or false(). If set to true(), then the function returns the count of *the visible row-groups* in the table; if set to false(), then the function returns the count of *all row-groups* (visible and invisible). (Visible row-groups are row-groups that have their Visible property set to **true**; *see here*[1078] *for a description of table properties*.) If the optional **VisibleOnly** argument is not submitted, then the count of *visible row-groups* is returned; the effect is the same as if submitting the **VisibleOnly** argument with a value of true().

**Note:** If the Visible property of a row-group has been defined, then a child row-group cannot use the **mt-table-rowgroup-count** function for visible row-groups.

⊟ *Examples*

- **mt-table-rowgroup-count**() returns **10** if the total number of row-groups in the current table is 10, all visible
- **mt-table-rowgroup-count**() returns **7** if, out of a total of 10 row-groups in the current table, seven are visible and three are invisible
- **mt-table-rowgroup-count**(true()) returns **7** if, out of a total of 10 row-groups in the current table, seven are visible and three are invisible
- **mt-table-rowgroup-count**(false()) returns **10** if, out of a total of 10 row-groups in the current table, seven are visible and three are invisible

## mt-table-rowgroup-index

**mt-table-rowgroup-index(VisibleOnly?** *as xs:boolean***) as xs:integer**
The function must be set inside a row-group, which becomes the context of the function. The function returns the position (or index) of the current row-group within the total number of row-groups of the current table. If the optional **VisibleOnly** argument is set to true(), then the function returns the index of the current row-group within the set of *visible row-groups* of the table; if set to false(), then the index among *all row-groups* (visible and invisible) is returned. (Visible row-groups are row-groups that have their Visible property set to **true**; *see here*[1078] *for a description of table properties*.) If the optional **VisibleOnly** argument is not submitted, then the index among visible row-groups is returned.

⊟ *Examples*

- **mt-table-rowgroup-index**() returns **1** if the current row-group is the first of 10 row-groups in the

current table, all row-groups visible
- `mt-table-rowgroup-index()` returns **1** if the current row-group is the second row-group and the first row-group is invisible
- `mt-table-rowgroup-index(true())` returns **1** if the current row-group is the second row-group and the first row-group is invisible
- `mt-table-rowgroup-index(false())` returns **2** if the current row-group is the second row-group and the first row-group is invisible

### mt-test-case-run

**`mt-test-case-run() as map(*)`**

Returns a map containing information about the currently executed test run [1403]. The map will contain the following `key:value` pairs: `"name":<the name of the test case>`, `"step":<the current step>`, `"count":<the total number of steps>`. If no playback is currently running, then then the map's key will contain empty values.

*Examples*

- `mt-test-case-run()` returns a simple map like `{"name":"MyTestCase", "step":"2", "count":"10"}`
- `mt-test-case-run()` returns a simple map like `{"name":"", "step":"", "count":""}`

### mt-text-to-speech-is-language-available

**`mt-text-to-speech-is-language-available(Language as xs:string) as xs:boolean`**

The *Language* argument can take string values of the form `en` (language code) or `en-US` (language-country code). If the language specified in the *Language* argument is available on the mobile device, then the function returns `true()`, otherwise `false()`.

*Usage*

`mt-text-to-speech-is-language-available("en")` returns `true()` if **en** or any **en-<country>** language variant is available on the mobile device, `false()` otherwise.
`mt-text-to-speech-is-language-available("en-US")` returns `true()` if **en-US** is available on the mobile device, `false()` otherwise.

### mt-text-to-speech-is-speaking

**`mt-text-to-speech-is-speaking() as xs:boolean`**

Returns `true()` if the playback of a Text to Speech action [727] is in progress, `false()` otherwise.

### mt-transform-image

*See the description of this function in the Image-related Functions [1718] section of the Altova Extension Functions.*

### mt-user-tried-to-cancel-actions

*Description*

If the user pressed the **Back** button or tried to exit the solution, the function returns `true()`. Otherwise `false()`is returned (the default value).

*Usage*

`mt-user-tried-to-cancel-actions()`

▼ mt-video-get-current-position

**mt-video-get-current-position(VideoControlName** *as xs:string***) as xs:integer**
Takes the name of a video control as its argument, and returns the current playback position (in seconds) of the video that is playing in this video control. If no video is playing in the control, an error is returned. Note that information about the current position is available only after playback has started, so the function should be used only subsequent to playback-start.

*Usage*
**mt-video-get-current-position**("Video-01") returns the current position of the video playing in the video control named `Video-01`.

▼ mt-video-get-duration

**mt-video-get-duration(VideoControlName** *as xs:string***) as xs:integer**
Takes the name of a video control as its argument, and returns the duration (in seconds) of the video that is playing in this video control. If no video is playing in the control, an error is returned. Note that information about duration is available only after playback has started, so the function should be used only subsequent to playback-start.

*Usage*
**mt-video-get-duration**("Video-01") returns the duration of the video playing in the video control named `Video-01`.

▼ mt-video-height

**mt-video-height(VideoControlName** *as xs:string***) as xs:integer**
Takes the name of a video control as its argument, and returns the height (in pixels) of the video that is playing in this video control. If no video is playing, an error is returned. Note that information about video height is available only after playback has started, so the function should be used only subsequent to playback-start.

*Usage*
**mt-video-height**("Video-01") returns the height of the video playing in the video control named `Video-01`.

▼ mt-video-is-playing

**mt-video-is-playing(VideoControlName** *as xs:string***) as xs:boolean**
Takes the name of a video control as its argument, and returns `true()` if a video is playing in this video control, `false()` otherwise.

*Usage*
**mt-video-is-playing**("Video-01") returns `true()` if a video playing in the video control named `Video-01`, `false()` otherwise.

▼ mt-video-width

**mt-video-width(VideoControlName** *as xs:string***) as xs:integer**
Takes the name of a video control as its argument, and returns the width (in pixels) of the video that is playing in this video control. If no video is playing, an error is returned. Note that information about video width is available only after playback has started, so the function should be used only subsequent to

playback-start.

*Usage*
**mt-video-width**("Video-01") returns the width of the video playing in the video control named `Video-01`.

▼ mt-wait-cursor-shown

*Description*
Returns `true()`if the client device is showing its Wait cursor, `false()` otherwise.

*Usage*
`mt-wait-cursor-shown()`

▼ mt-zebra-scanner-connected

*Description*
Returns `true()` if a Zebra scanner is currently connected to the solution through one of the actions listed below, `false()` otherwise.

- Zebra Connect/Disconnect 772
- Zebra Mobile Connect/Disconnect 775

*Usage*
`mt-zebra-scanner-connected()`

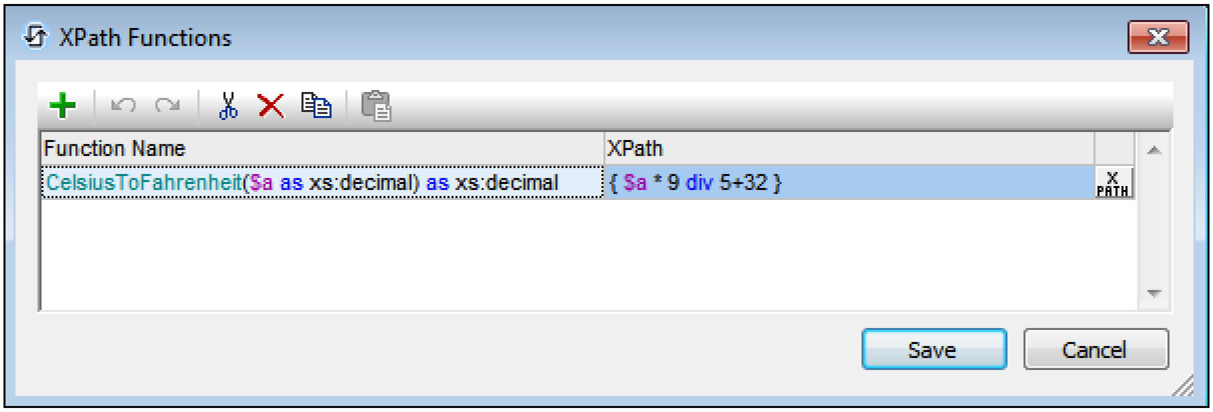
▼ mt-zebra-scanner-id

*Description*
Returns the ID of the currently connected Zebra scanner, or returns `-1` if no Zebra scanner is currently connected.

*Usage*
`mt-zebra-scanner-id()`

# 13.3 User-Defined XPath/XQuery Functions

You can create your own custom-made XPath/XQuery functions for individual projects, which you can then use in all XPath expressions in the project. The access point for creating and managing these user-defined functions is the XPath Functions dialog, accessed with the command, **Project | XPath/XQuery Functions**1591. The XPath Functions dialog (*screenshot below*) lists all the user-defined XPath functions in the project. You can add and delete functions using the corresponding icons in the toolbar of the dialog. To edit the definition of a function, click the function's **Edit XPath Expression** button.



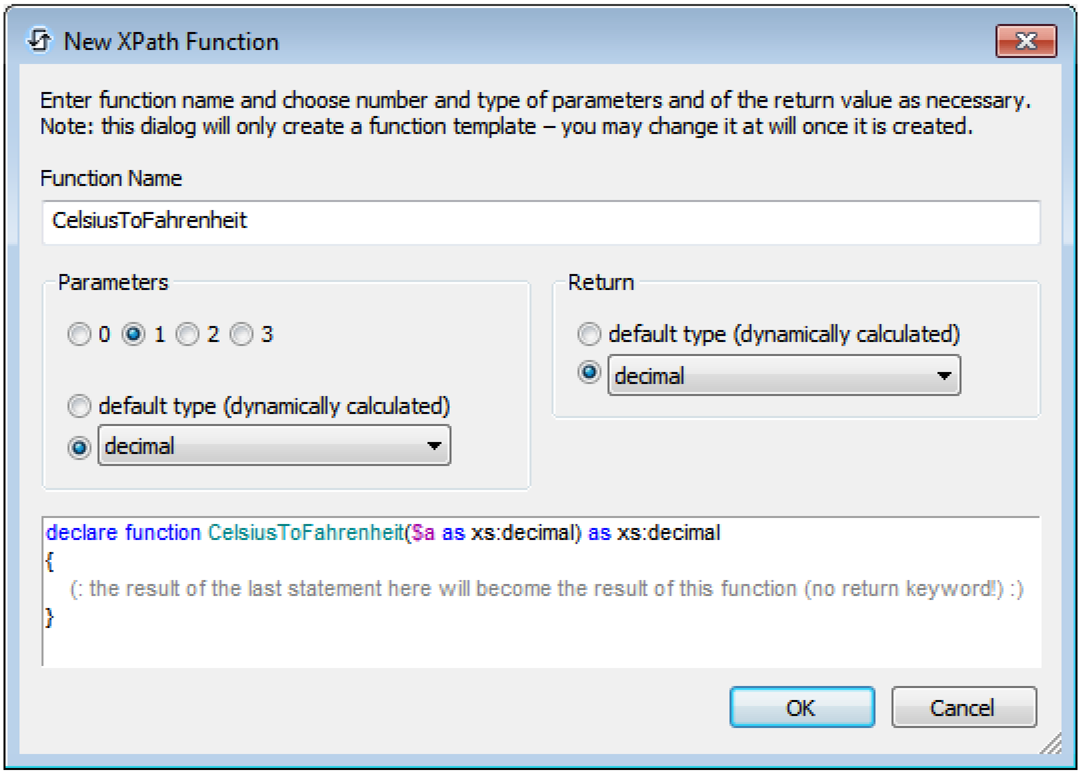
| Function Name | XPath |
|---|---|
| CelsiusToFahrenheit($a as xs:decimal) as xs:decimal | { $a * 9 div 5+32 } |

The function list can be ordered by function name. Do this by clicking the header of the Function Name column. Each click cycles the ordering through the following ordering sequence: (i) ascending, (ii) descending, (iii) dialog order. The order in the dialog can be modified by dragging-and-dropping functions to other positions in the list. Note that, if you order the list in ascending/descending order and then move a function to a different position in the list, the newly created order becomes the new dialog order.

## Add a new user-defined XPath function

Adding a new user-defined function involves two steps: (i) declaring the function, and (ii) defining the function.

To add a new function, do the following, click **Add** in the toolbar of the dialog (*see screenshot above*). This displays the New XPath Function dialog (*screenshot below*).



Enter function name and choose number and type of parameters and of the return value as necessary.
Note: this dialog will only create a function template – you may change it at will once it is created.

Function Name: CelsiusToFahrenheit

Parameters: ○ 0 ● 1 ○ 2 ○ 3

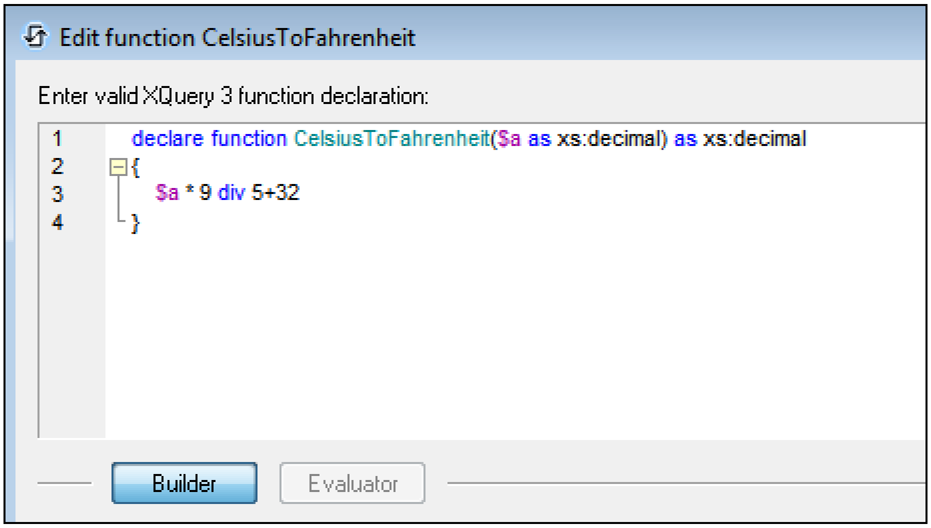
○ default type (dynamically calculated)
● decimal

Return:

○ default type (dynamically calculated)
● decimal

```
declare function CelsiusToFahrenheit($a as xs:decimal) as xs:decimal
{
    (: the result of the last statement here will become the result of this function (no return keyword!) :)
}
```

In this dialog, you can declare the name of the function, specify the number of function parameters (arguments) and their types, and specify the return type of the function. In the screenshot above we have declared a function to convert a decimal number from Celsius to Fahrenheit. The function takes one parameter, which is the input Celsius value as a decimal. It will output a decimal value, the Fahrenheit temperature. What the function does is defined in the next step. After declaring the function (*screenshot above*), click **OK**. This displays the Edit Function dialog (*screenshot below*), which contains the template of the newly declared function and in which you can now define the function.


Edit function CelsiusToFahrenheit
Enter valid XQuery 3 function declaration:
Builder
Evaluator

```
declare function CelsiusToFahrenheit($a as xs:decimal) as xs:decimal
{
    $a * 9 div 5+32
}
```

Enter the definition of the function within the braces. In the definition shown in the screenshot above, **$a** is the input parameter. Click **OK**. when done. The function will be added to the list of user-defined functions in the XPath Functions dialog and can be used in all XPath expressions in the project.

**Note:** User-defined XPath functions do not need to be placed in a separate namespace. Consequently, no namespace prefix is needed when defining or calling a user-defined function. The XPath default namespace[310] is used for all XPath/XQuery functions, including extension functions and user-defined functions[1293]. In order to avoid ambiguities involving built-in functions, we recommend that you capitalize user-defined functions.

## 13.4 FAQ about XPath/XQuery

***I have an XPath expression inside the repeating row of a table. Should I use an absolute or relative XPath expression to target a child attribute?***

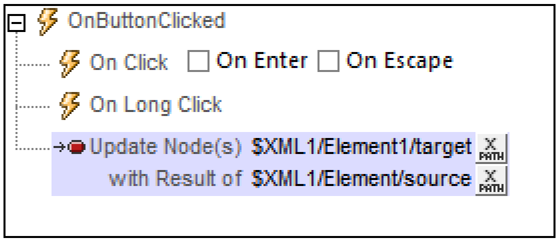
If your XPath expression is inside the repeating row of a table, then the element corresponding to the row of the table is the context node, say, `Row`.

- If you use an absolute path, for example `$XML/Row/@id`, then the XPath expression will return a sequence of the `@id` values of **all** `Row` elements. If you are using an operation that expects one atomic value, the operation will generate an error.
- If you use a relative path, for example `@id`, then, since for every repeating row, you have a context `$XML/Row`, the XPath expression will correctly return the single atomic value of the one `@id` attribute of the current `Row` element.

***I have an XPath expression that locates a mixed-content element (text and element). Instead of getting the text value of the located element and its descendants (as mandated by XPath), I get the text content of the located element only. Why?***

If an element with mixed content (text and element/s) is located with an XPath locator expression, then the text content of the mixed-content element only is returned. The text content of descendant elements is ignored.

This is best explained with an example of the Update Node(s) action[886]. Consider the Update Node(s) action[886] defined in the screenshot below.



OnButtonClicked
On Click □ On Enter □ On Escape
On Long Click
Update Node(s) $XML1/Element1/target
with Result of $XML1/Element/source
XPATH
XPATH

If the XML tree had the following structure and content:

```
<Element1>
   <source>AAA
      <subsource>BBB</subsource>
  </source>
  <target></target>
</Element1>
```

Then the `target` element would be updated with the text content of the mixed-content element `source`, while ignoring the content of its child element `subsource`. The node named `target` will be updated to `<target>AAA</target>`.

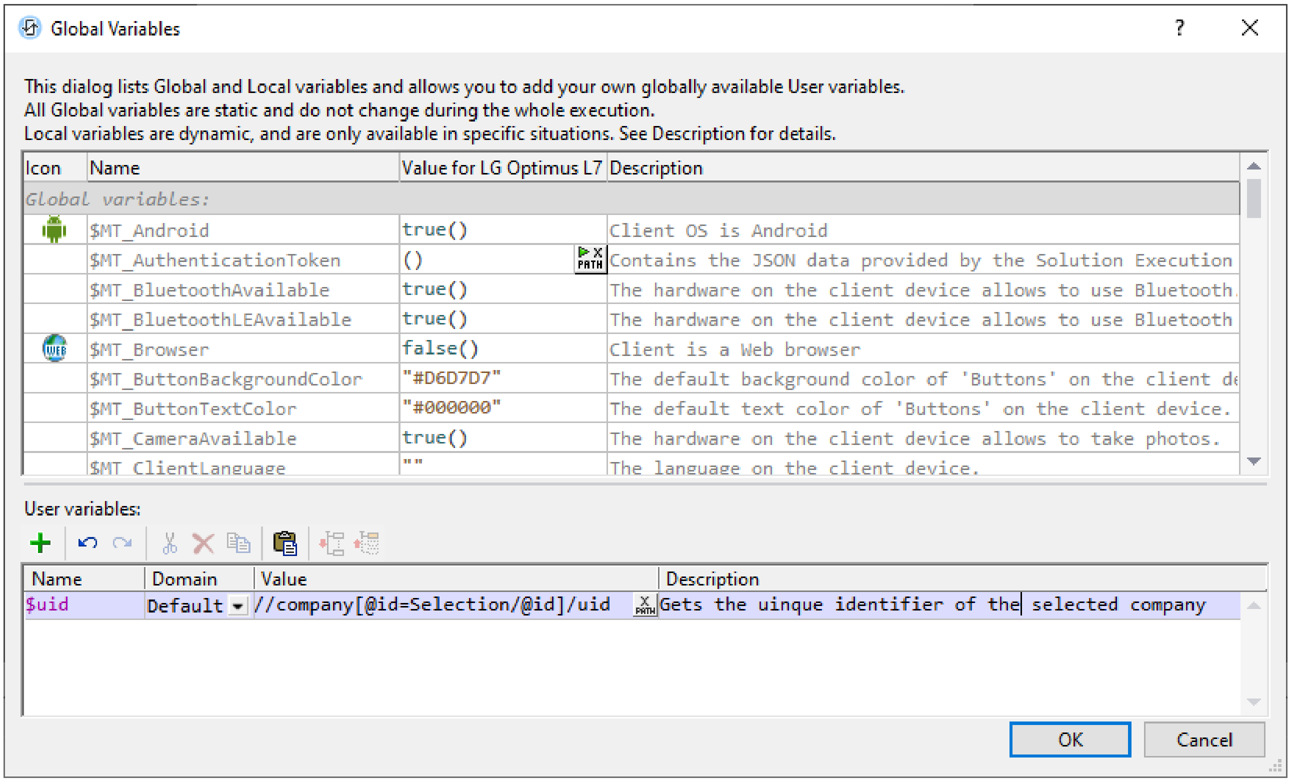
**Note:** If you wish to include the text content of descendant node/s, use a **`string`** function. Using the XML example above, for instance, the expression `string($XML1/Element1/source, '')` will return `"AAABBB"`.

**Note:** Charts use the XPath compliant method of serializing: When a mixed-content element is located using an XPath locator expression, then the text content of descendant elements is also serialized.

# 14 Global Variables

Global variables contain information about the client mobile device. For example, there is one variable to indicate the device's type, another to indicate its dimensions, and yet another to indicate the device's current orientation (landscape or portrait), and so on. The values of all these variables are obtained at run-time from the client device as part of standard mobile communication procedures. Variables can then be used in XPath/XQuery expressions. As a result, processing can be specified that is conditional upon a device's inherent static properties (such as size) or its changeable dynamic properties (such as orientation).

MobileTogether Designer has a standard library of global variables, which is displayed in the Global Variables dialog (**Project | Global Variables**, *screenshot below*). In this dialog, you can also define custom variables for use throughout the project. The values of customized user variables are set with XPath expressions.



This dialog lists Global and Local variables and allows you to add your own globally available User variables.
All Global variables are static and do not change during the whole execution.
Local variables are dynamic, and are only available in specific situations. See Description for details.

| Icon | Name | Value for LG Optimus L7 | Description |
|---|---|---|---|
| | *Global variables:* | | |
| | $MT_Android | true() | Client OS is Android |
| | $MT_AuthenticationToken | () | Contains the JSON data provided by the Solution Execution |
| | $MT_BluetoothAvailable | true() | The hardware on the client device allows to use Bluetooth. |
| | $MT_BluetoothLEAvailable | true() | The hardware on the client device allows to use Bluetooth |
| WEB | $MT_Browser | false() | Client is a Web browser |
| | $MT_ButtonBackgroundColor | "#D6D7D7" | The default background color of 'Buttons' on the client de |
| | $MT_ButtonTextColor | "#000000" | The default text color of 'Buttons' on the client device. |
| | $MT_CameraAvailable | true() | The hardware on the client device allows to take photos. |
| | $MT_ClientLanguage | "" | The language on the client device. |

User variables:

| Name | Domain | Value | Description |
|---|---|---|---|
| $uid | Default | //company[@id=Selection/@id]/uid | Gets the uinque identifier of the selected company |

The Global Variables dialog (*screenshot above*) displays three types of variables:

- Static-value variables [1300]: These variables contain values that do not change during the execution of the project. Notice that the header of the *Value* column indicates the mobile device that has been selected in the Device Selector combo box [254]. The values of variables vary with the client device. For example, the variable `$MT_Android` has a value of **`true()`** when the mobile device being used is an Android.
- Dynamic-value variables [1304]: These variables contain device-related and project-related values that can change during execution. For example, the `$MT_ControlNode` variable has different values according to which node is the current node at a given time during project execution.

- User variables[1309]: In addition to the standard library of global variables, you can add your own global variables (called *User Variables* in the dialog) in the lower pane of the dialog. You use XPath expressions to give a user variable a value.

**Note:** When defining a user variable, do not use a **$** symbol in the name of the variable. When you use any global variable in an XPath expression, however, you must, as usual, use the **$** symbol. For example:

```
concat('http://www.', $company, '.com')
```

## 14.1 Static Global Variables

Static-value variables are called *Global Variables* in the Global Variables dialog[1298]. They are variables that contain static information about the mobile device, such as the device's type and dimensions. Values of static variables do not change during the execution of the project. They are displayed in the Global Variables dialog[1298] (**Project | Global Variables**[1590]). In the dialog, the header of the *Value* column displays the mobile device that is selected in the Device Selector combo box[254]. For example, the variable `$MT_Android` has a value of **`true()`** when the mobile device being used is an Android device. (Device information is sent by the device as part of standard mobile communication procedures.)

**Note:** Please see the Global Variables[1298] dialog for a complete list of variables and their descriptions.

▼ Variables that indicate the mobile-device type

*Description*

These are a set of variables (*see table below*) that indicate the device type. They can be used to specify actions that are conditional on the device type. For example: `if ($MT_iOS=true()) then 'http://www.apple.com/' else 'http://www.altova.com'`. Information about the client device is sent by the device. If the solution runs on a particular device, then the corresponding global variable (*see table below*) is set to **`true()`**; all the other variables in the group are set to **`false()`**. All these variables can then be used in XPath/XQuery expressions.

| | |
|---|---|
| **MT_Android** | true() \| false() |
| **MT_Browser** | true() \| false() |
| **MT_iOS** | true() \| false() |
| **MT_iPad** | true() \| false() |
| **MT_Windows** | true() \| false() |
| **MT_WindowsPhone** | true() \| false() |

▼ Variables that indicate the device's communications capability

*Description*

These variables indicate whether Bluetooth, SMS, and telephony services are available on the mobile device. These variables can be used to make checks before initiating Bluetooth actions[770] or SMS or call actions[671]. Information about the communications capability is received from the client device. Values can be **`true()`** or **`false()`**. If these functions are not available (for example, when the client is a web browser) then these variables are undefined (the empty string).

| | |
|---|---|
| **MT_BluetoothAvailable** | true() \| false() \| "" (empty string) |
| **MT_BluetoothLEAvailable** | true() \| false() \| "" (empty string) |
| **MT_SMSAvailable** | true() \| false() \| "" (empty string) |
| **MT_TelephonyAvailable** | true() \| false() \| "" (empty string) |

▼ Variables that indicate the availability of device features

*Description*

These variables indicate whether device features, such as a camera application or geolocation tracking, are available on the mobile device. They can be used to make checks before initiating image-taking[1090], geolocation-related[733], NFC-related[744], or barcode scanner[770] actions. Information about the feature availability is received from the client device. Values can be **`true()`** or **`false()`**. If these functions are not available (for example, when the client is a web browser) then these variables are undefined.

| | |
|---|---|
| **`MT_CameraAvailable`** | `true() \| false() \| undefined` |
| **`MT_DatalogoicScannerAvailable`** | `true() \| false() \| undefined` |
| **`MT_GeolocationAvailable`** | `true() \| false() \| undefined` |
| **`MT_NFCAvailable`** | `true() \| false() \| undefined` |
| **`MT_ZebraMobileComputerAvailable`** | `true() \| false() \| undefined` |

▼ Variables that contain the device's dimensions and resolution

*Description*

The absolute height and width of the device's display are held by these variables as pixel values. The resolution is in terms of dpi (pixels per inch), in the X and Y dimensions. The `$MT_DPIX` and `$MT_DPIY` variables for iOS devices are empty.

| | |
|---|---|
| **`MT_DeviceHeight`** | *Length value in pixels* |
| **`MT_DeviceWidth`** | *Length value in pixels* |
| **`MT_DPIX`** | *Horizontal pixel density in pixels per inch* |
| **`MT_DPIY`** | *Vertical pixel density in pixels per inch* |

▼ Variables that contain default colors of device elements

*Description*

Pages and some page controls have different default colors on different devices. Knowing the default colors is useful for designing the look of the page. For example, a label's background color can be set conditionally according to what the default label text color on the device is: `if (`**`$MT_LabelTextColor = '#000000'`**`) then '#FFFFFF' else '#000000'`. The default colors are received from the client device and are hexadecimal values, e.g: `#336699` and `#ffaaff`.

| | |
|---|---|
| **`MT_ButtonBackgroundColor`** | *Background color of buttons; Hex values, e.g:* `#ffaaff` |
| **`MT_ButtonTextColor`** | *Text color of buttons; Hex values, e.g:* `#336699` |
| **`MT_EditFieldBackgroundColor`** | *Background color of edit fields; Hex values, e.g:* `#ffaaff` |
| **`MT_EditFieldTextColor`** | *Text color of edit fields; Hex values, e.g:* `#336699` |
| **`MT_LabelBackgroundColor`** | *Background color of labels; Hex values, e.g:* `#ffaaff` |
| **`MT_LabelTextColor`** | *Text color of labels; Hex values, e.g:* `#336699` |
| **`MT_PageBackgroundColor`** | *Background color of pages; Hex values, e.g:* `#ffaaff` |

▼ Miscellaneous

⊟ MT_AuthenticationToken

The Solution Execution 915 provides a setting for passing an authentication token to a web page when the solution is executed on web clients. This variable holds the authentication data sent by the action.

⊟ MT_ClientLanguage

The language on the client device.

⊟ MT_InputParameters

Parameter values are passed to the solution when the solution is started. These values are stored in the **`MT_InputParameters`** variable. By default the data structure that is stored in the variable is a map (for example: `{"name":"Altova", "location":"Boston"}`). You can change the data structure of this variable in individual projects (in the More Project Settings dialog 296) to be a sequence of values (for example: `("Altova", "Boston")`).

Currently, parameter values are passed to the solution when a hyperlink to the solution is clicked or when the **`OnServerDeployment`** 296 event is triggered. If the hyperlink's URL has a query string that contains parameter values, then these are passed to the solution when the link is clicked and the solution is started. The parameter values in the query string must be in **`key:value`** format.

The **`MT_InputParameters`** variable holds parameter values either (i) as a map, or (ii) as a sequence of string-value items that are alphabetically sorted on the query keys. If the values are stored in a map, then the **`key:value`** pairs are stored. If the values are stored as a sequence of strings, then the strings are indexed alphabetically on their keys. The latter case can be explained with an example, Say the query has three keys. When the keys and their are received by the solution, the keys will be sorted alphabetically and their respective values will be stored as the corresponding items, by index position, in the **`MT_InputParameters`** sequence of string values. To retrieve a single parameter value from the sequence of values, you must know that parameter's index position in the sequence. You can then use this position in an XPath locator expression, for example: **`$MT-InputParameters[1]`**. will return the first item in the sequence of string values. For more information about hyperlinking and the `MT_InputParameters` variable, see Hyperlinking to Solutions 1239.

⊟ MT_IsAppStoreApp

Indicates whether the current solution is running as an AppStore App 1471 or not. Allowed values are `true()` or `false()`, with the default being `false()`.

⊟ MT_IsEmbedded

Indicates whether the current solution is running embedded in a webpage 1427 or not. Allowed values are `true()` or `false()`, with the default being `false()`.

⊟ MT_SimulationMode

Indicates, using the values listed in the table below, the kind of simulation that is currently running. The empty-sequence value indicates that the solution is running in an actual end-user scenario, and not in a simulation. `$MT_SimulationMode` is useful, for example, if you want to provide conditional processing depending on what kind of simulation (or actual use) is currently running. See the section Simulation 1355 for more information.

| | |
|---|---|
| **`"designer"`** | *Simulation runs directly in designer* |
| **`"designer-server"`** | *Simulation with a standalone server* |
| **`"designer-client"`** | *Simulation is a trial run on client* |
| **`()`** | *Server to client/browser, run by end user* |

⊟ MT_UserName

The name with which to log in to MobileTogether Server.

## 14.2 Dynamic Local Variables

Dynamic-value variables are called *Local Variables* in the Global Variables dialog 1298. They contain device-related and project-related information that can change during project execution. For example, the device orientation variables will change according to how the end user is currently holding the device (*see the description of the device orientation variables below*).

The variables that contain information about the current control (*see below*) are particularly useful because they can be used to refer to different aspects of the control and the node being currently processed. Being able to identify the current control and node enables conditional processing. For example, the `$MT_ControlNode` variable can be used to test which node is the current node at a given time during project execution, and locate another node on this basis. The `$MT_ControlValue` variable holds the content of the node associated with the current control.

**Note:** Please see the Global Variables 1298 dialog for a complete list of variables and their descriptions.

▼ Variables that indicate device orientation

*Description*

The values of `MT_Portrait` and `MT_Landscape` can be **`true()`** or **`false()`**and can change during the course of project execution. They can be used to specify page or control properties according to device orientation.

| | |
|---|---|
| **`MT_Portrait`** | `true() \| false()` |
| **`MT_Landscape`** | `true() \| false()` |

▼ Variables that indicate the device's viewport dimensions

*Description*

These variables provide, respectively, the width (X dimension) and height (Y dimension) of the device's viewport. Note that the value of the X dimension changes with the orientation (portrait/landscape), as does the Y dimension. The viewport is the screen area on which design components are drawn; it is the screen area minus the top and/or bottom bars that hold tabs/buttons. In web-browser clients, the `$MT_CanvasX` and `$MT_CanvasY` variables give the dimensions of the canvas on which the MobileTogether Client app is displayed (that is, the dimensions of the browser window minus the title bar, ribbon, status bar, and any sidebars). The values of these variables are pixel values and will necessarily be less than the device's height and width dimensions (returned respectively by **`MT_DeviceHeight`** 1300 and **`MT_DeviceWidth`** 1300). See the note *Points Versus Pixels on iOS devices* below.

| | |
|---|---|
| **`MT_CanvasX`** | *Width, as a length value in pixels* |
| **`MT_CanvasY`** | *Height, as a length value in pixels* |

⊟ *Points versus pixels on iOS devices*

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** 1304 and **$MT_CanvasY** 1304 dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Variables that contain the size of resizable windows (in Windows apps and browsers)

*Description*
These variables are applicable only for for web browsers and Windows app devices. Browser windows, as well as app windows on Windows RT devices and touch-enabled Windows operating systems, can be re-sized by the user (just like windows in desktop apps can be re-sized). The `$MT_WindowHeight` and `$MT_WindowWidth` variables hold the height and width, respectively, of the window in which the MobileTogether Client app is running. In browsers, these variables give the height and width of the browser window. (In browsers, the canvas on which the MobileTogether Client app is displayed (that is, the dimensions of the browser window minus the title bar, ribbon, status bar, and any sidebars) is given by the `$MT_CanvasX` and `$MT_CanvasY` variables).

| | |
|---|---|
| **MT_WindowHeight** | *A length value in pixels* |
| **MT_WindowWidth** | *A length value in pixels* |

▼ Variables that contain information about the current control

*Description*
These variables contain information related to the current control and its associated page source node (the source node 347 of the control). The values of these variables change during execution according to which control is being currently processed. For example, the `$MT_ControlNode` variable has different values as the associated node changes as the current control changes. (Note that some controls, such as the space and horizontal line controls, do not have page source links, while others, like the chart control, will not have an XML value as the content of its associated node.)

The `$MT_ControlNode` variable is a pointer to the source tree node. So you can use it for tests such as this: **`$MT_ControlNode/localname()="first"`**.

These variables are useful for changing a control's properties based on the control's values. For example, a `$MT_ControlValue` variable can be used to change a label's background color to red in case an error is encountered: `if (`**`$MT_ControlValue = 'NaN'`**`) then '#FF0000' else '#FFFFFF'`.

| | |
|---|---|
| **MT_ControlKind** | *The kind of control, as a string. Example: "Label"* |

| | |
|---|---|
| **`MT_ControlName`** | *The name you give the control, as string.*<br>*Example: "Label-1"* |
| **`MT_ControlNode`** | *XML node that is the control's source node* |
| **`MT_ControlValue`** | *Value of the control's page-source-link node* |
| **`MT_ControlValueBeforeChange`** | *Previous value of the control's page-source-link node, before the control or node is edited* |

**Note:** The `$MT_ControlValue` variable is **not** available for the generation of the values of the `Visible`, `Get Value From XPath`, and `Text` properties of controls[404]. If used for the values of these properties, then a validation error results.

▼ Miscellaneous

⊟ MT_AudioChannel

This variable is usable only in actions that are defined for Audio playback events[1109]. It holds an integer that is the number of the audio channel (`1 to 5`) that triggered the event.

⊟ MT_Broadcast

If a solution is subscribed to a broadcast topic and receives messages, then the latest received message is stored in this variable. The message can then be accessed via this variable, typically in the page event `OnBroadcastReceived` or the project event `OnBroadcastReceived`.

⊟ MT_DBExecute_Result

The XML result of the last executed DB Execute action[861]. Note that any kind of SQL statement can be used in the DB Execute action[861]. So executing the action could obtain various kinds of result XML data, including, for example, data from the DB, boolean values, or computational results.

⊟ MT_DragAndDropSource

Contains the context node of the source node to be dropped. This is the the row group of the context node. See Table Properties[1078] for information.

⊟ MT_DragAndDropTarget

Contains the context node of the target, which is the row group in which the source is to be dropped. See Table Properties[1078] for information.

⊟ MT_FirstPageLoad

Set to `true()` if this is the first time the page[257] is loaded during the current workflow execution.

⊟ MT_GeolocationMapMarker

The dynamic variable **`$MT_GeolocationMapMarker`** contains information about the marker that was last clicked by the client's user. This information that is contained in the variable is stored in an XPath-map construct, in a format that is shown by the following example:

```
map {
     "id":"vie",
     "geolocation":(48.2143531, 16.3707266),
     "title":"Vienna",
     "text":"Altova EU"
}
```

To fetch a value from the XPath-map construct, use an XPath expression like this: `map:get( $MT_GeolocationMapMarker, "id" )`. This particular expression returns the value of the **`id`** key (that is, the **`id`** of the marker that was clicked).

### MT_HTTPExecute_Result

The XML result of the last executed Execute SOAP Request[835] or Execute REST Request[837].

### MT_MeasureControls

Automatically stores the result of the last-executed Measure Controls[927] action.

### MT_PageName

The name of the page[257].

### MT_Progress

Contains the data sent by the server via the Progress Update[796] action. The data that is sent is the value of the Progress Update[796] action's *Value* parameter. See the Progress Indicator tutorial[241] for an example of how to use this variable.

### MT_ServerConnectionErrorLocation

This variable is a sequence of strings that contains the action stack that triggered the `OnServerConnectionError`[389] event. Since action names can change from release to release, this variable should be used only for debugging.

### MT_TableColumnContext

This variable contains the context node of the current column when a table with dynamic columns is being generated. It is indispensable when working with tables that contain both dynamic rows as well as dynamic columns. In such tables, cell content is defined in the context of the element that is associated with the dynamic row. Within this row context, the **`MT_TableColumnContext`** variable can be used to locate the element that is associated with the current column. For an example of how to use the variable, see the section Tables | Dynamic Columns[1074].

### MT_TargetNode

This variable identifies the target node of an Update Node(s)[886], Insert Node(s)[880], Append Node(s)[875], or Replace Node(s)[883] action. It can be used to generate update values and new node properties according to what the target node is. See the descriptions of the respective actions for examples of how the variable can be used. `$MT_TargetNode` can also be used with the DB Execute[861] action and the **Ensure Exists on Load (XPath)**[371] command.

### MT_UpdatedInAppPurchases

Contains a sequence of SKU-IDs, as strings, of in-app purchases[1502] that have been updated in the In-App-Purchase Page Source[1507]. The variable is in scope only while actions of the **`OnPurchaseUpdated`**[1513] event are being processed.

⊟ MT_UserMail

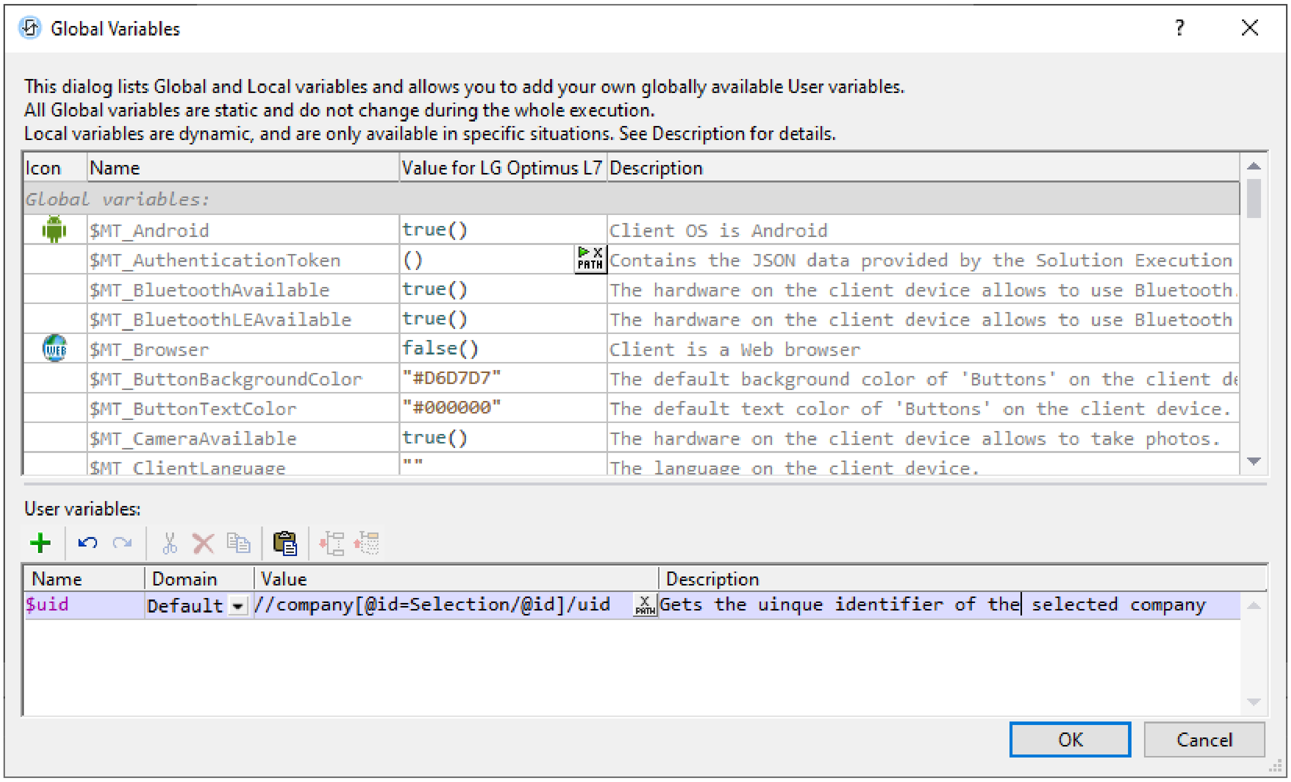
The email address to which messages from MobileTogether Server will be sent. These messages from the server typically relate to administrative events and tasks on the server, and the email address of the recipient of these messages is typically specified in the MobileTogether Server settings. The **`MT_UserMail`** variable, however, enables you to specify the email address via the design.

⊟ MT_UserRoles

The roles of the currently logged in user. The roles are those that are assigned to the user by the MobileTogether Server administrator, and are obtained from MobileTogether Server.

# 14.3 User Variables

User Variables are variables that you define in the lower pane of the Global Variables dialog (**Project | Global Variables** 1590, *screenshot below*). They are very useful if you wish to store data that can be accessed by multiple objects at different times during execution.



Global Variables

This dialog lists Global and Local variables and allows you to add your own globally available User variables.
All Global variables are static and do not change during the whole execution.
Local variables are dynamic, and are only available in specific situations. See Description for details.

| Icon | Name | Value for LG Optimus L7 | Description |
|---|---|---|---|
| *Global variables:* | | | |
| | $MT_Android | true() | Client OS is Android |
| | $MT_AuthenticationToken | () | Contains the JSON data provided by the Solution Execution |
| | $MT_BluetoothAvailable | true() | The hardware on the client device allows to use Bluetooth. |
| | $MT_BluetoothLEAvailable | true() | The hardware on the client device allows to use Bluetooth |
| | $MT_Browser | false() | Client is a Web browser |
| | $MT_ButtonBackgroundColor | "#D6D7D7" | The default background color of 'Buttons' on the client de |
| | $MT_ButtonTextColor | "#000000" | The default text color of 'Buttons' on the client device. |
| | $MT_CameraAvailable | true() | The hardware on the client device allows to take photos. |
| | $MT_ClientLanguage | "" | The language on the client device. |

User variables:

| Name | Domain | Value | Description |
|---|---|---|---|
| $uid | Default | //company[@id=Selection/@id]/uid | Gets the uinque identifier of the selected company |

To add a user variable, in the lower pane, do the following:

1. Click the **Append** or **Insert** icons (located in the pane's toolbar) to add a new entry to the list.
2. Enter the name of your new variable (in the *Name* column, without a **$** symbol) and give the variable a description (*Description* column). *See screenshot above*.
3. Click in the *Value* field to bring up the Edit XPath/XQuery Expression dialog 1244, and enter the XPath expression that determines the value of the variable.
4. In the *Domain* field, you can choose whether to store the variable on the client only, the server only, or on both (this is the default). The *Client Only* option is useful if the variable contains or involves a large dataset and you want to avoid possible performance slowdowns that might occur when data is transferred between client and server.
5. Select an icon to help identify the new variable as belonging to a particular group.
6. Click **OK** to finish. The variable is added as a global variable, and can be used in programming contexts.

## Variables to store styling property values

You can store the value of a styling property in a variable: as a string that exactly matches a valid CSS property value. For example, you could set a variable named `MyTextColor` and give it a value of `"#AA6633"`.

You could then use the variable **`$MyTextColor`** as the property of `Text Color` wherever desired in the design. A set of commonly used properties can be specified via variables, and the styles they define will be displayed not only at run time and during simulations, but also in the design itself.

This feature (styling via variables) is currently provided for the following style properties:

- All color styles (text, background, etc)
- Line styles (color, style, width)
- Text styles (size, weight, etc)
- All paddings and margins
- Borders
- Vertical and horizontal alignment
- Table column width, maximum column width
- Control width and height

Note the following points:

- Where units need to be specified, write the units correctly. For example, **`"16px"`**. If you have any doubt about the correct unit to use or about the way to write the unit, look up that property's available units in the Styles & Properties Pane 274.
- The only supported length unit is pixels (**`px`**).
- The value of the variable can only be a literal value, expressed as a single string.
- If you use an XPath expression that uses operators, constructs, or functions (for example, an **`if...then...else`** construct), then the variable that uses such a construct cannot be used, but it can be passed to a styling variable. The styling variable will work if the first variable has been evaluated already (typically in the static context) and if it contains a string when it is passed to the styling variable.
- The XPath expressions of user-defined variables are re-evaluated only in situations where global variable values can change, for example, when the orientation of the client device is changed.

# 15 Presentation

Presentation refers to the display of pages and their components on client devices. It includes (i) the layout of the page components relative to each other and (ii) the display properties of individual components. The page display model closely follows the CSS model, and presentation properties include those for setting margins, padding, background colors, text colors, and fonts of individual page components. You can therefore set presentation properties on the page as well as the controls of the page.

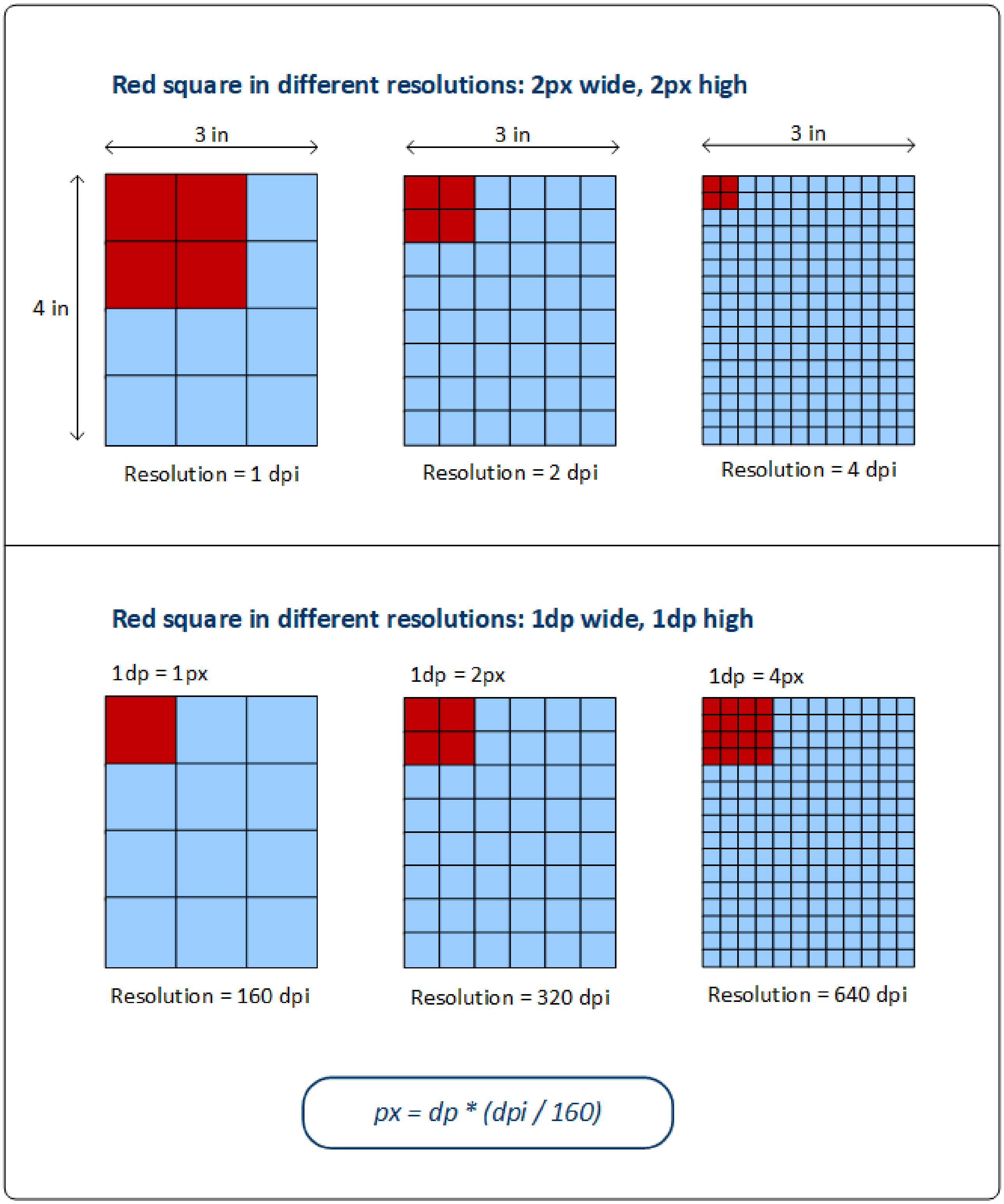
This section is organized int the following subsections:

- Sizes: Pixels, DPI, DP, SP 1312, which provides information about the size units used in MobileTogether
- How to Set Styles 1315, which lists the various places in the design in which styles can be set and how they interact with each other
- Style Sheets 1318, which describes the style sheet feature of MobileTogether
- Style Variance Across Clients 1330, which lists default display rules on platforms that are different from expected behavior and how you can compensate for these

## 15.1 Sizes: Pixels, DPI, DP, SP

The size of objects and text in the design can be specified in pixels (px). However, the display on the client device depends not only on the specified pixel size but also on the device resolution, and, in the case of text, additionally on the text size selected by the device user. *See the first row of the diagram below.*

The new units of **device-independent pixels (dp)** and **scale-independent pixels (sp)**, however, enable you to obtain a display of the same size across devices that have different resolutions. The diagram below shows how resolution affects pixel sizes and dp sizes.


Red square in different resolutions: 2px wide, 2px high
3 in
3 in
3 in
4 in
Resolution = 1 dpi
Resolution = 2 dpi
Resolution = 4 dpi
Red square in different resolutions: 1dp wide, 1dp high
1dp = 1px
1dp = 2px
1dp = 4px
Resolution = 160 dpi
Resolution = 320 dpi
Resolution = 640 dpi
px = dp * (dpi / 160)

*Pixels and resolution*
The resolution of a screen is the number of pixels in an inch of screen length; its unit is **`ppi`** (pixels per inch), more commonly referred to as **`dpi`** (dots per inch). So if the pixel density of a screen (its dpi) is high, then an object with the same pixel size will be displayed smaller on higher resolution screens. This can lead to the same object being displayed in different sizes on screens that have the same dimensions but different resolutions. In the top row of the diagram above, for example, all the rectangles (representing devices) have the same width and height, but different resolutions. As a result, the red square (`width=2px` and `height=2px`) is progressively smaller in the higher resolution rectangular screens. For example, the rightmost screen has a resolution of 4 pixels per inch; as a result, the red area of 2x2 pixels covers a smaller area of this screen. This difference in area size can be overcome by using device-independent pixels (`dp`) as the unit of length.

*Device-independent pixels (dp)*
If device-independent pixel (`dp`) is used as the unit of length, then the operating system of the device maps the dp value to a corresponding number of pixels based on the resolution of the device screen. For this mapping, 1 dp is considered to be equal to 1 pixel on a 160 dpi resolution screen. The corresponding number of pixels can be calculated with the formula `px = dp * (dpi/160)`. By using device-independent pixels, you will have better control over your design across devices of differing resolutions.

*Scale-independent pixels (sp)*
A scale-independent pixel (`sp`) is the same as a device-independent pixel (`dp`), with an additional scaling factor that is based on the font size that the user selects in the device's system settings. Scale-independent pixels should be used only as a unit for text. Avoid using `sp` as a unit for non-text components.

**Note:** DP and SP sizes in the designer's device simulations 1355 will not exactly match sizes on the actual client devices.

## XPath extension function: mt-convert-units

MobileTogether has a built-in XPath extension function, which you can use for converting between the three values:

▼ mt-convert-units

**`mt-convert-units(Size`** *`as xs:string`*, **`TargetUnit`** *`as xs:string`*`)` **`as xs:string`**
Converts the length value specified in the **`Size`** argument to an equivalent value in the unit specified by the **`TargetUnit`** argument. You can convert between any two of the following units: **`px`**, **`dp`**, and **`sp`**. Both input arguments, as well as the output value, are strings. For more information about units and conversion between them, see Sizes: Pixels, DPI, DP, SP 1312.

⊟ *Examples*

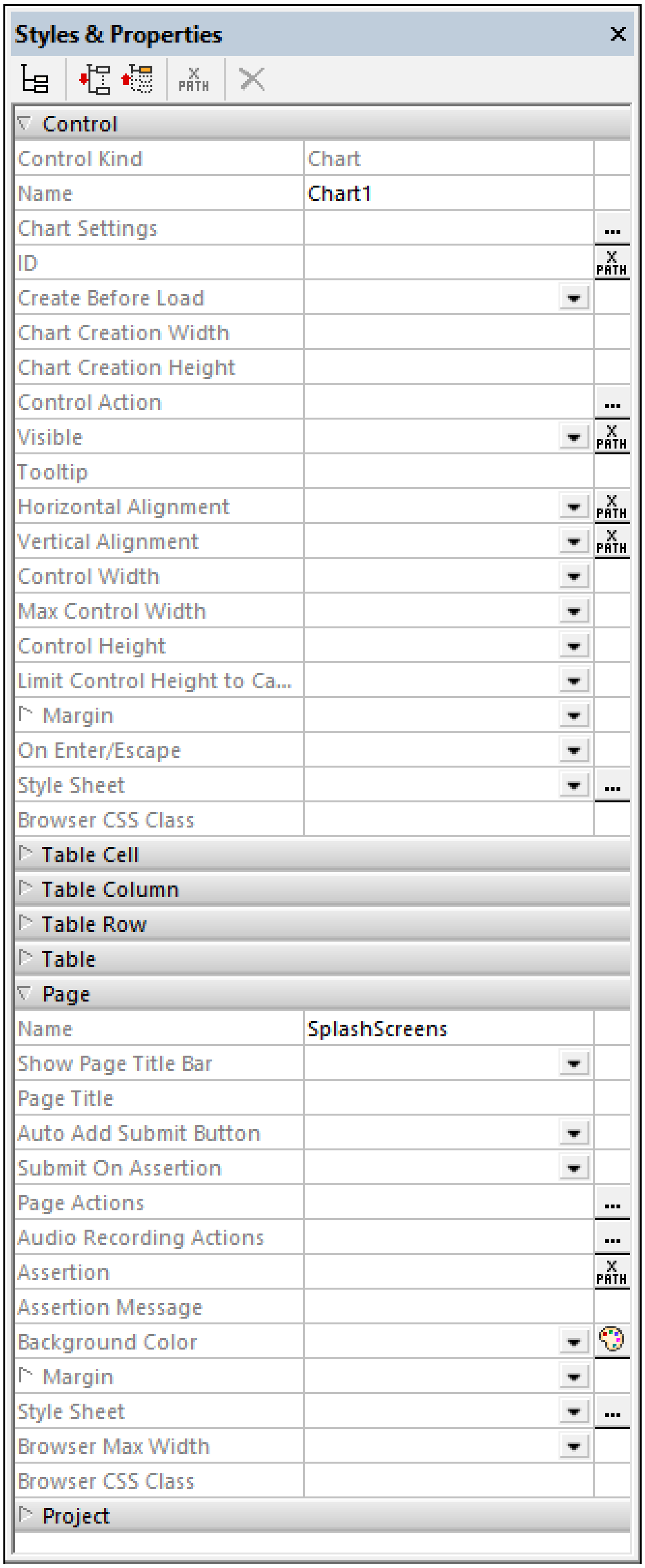
- **`mt-control-unit("24px", "dp")`** returns, depending on the device's resolution, for example, **`"22dp"`** on one device and **`"20dp"`** on another
- **`mt-control-unit("20sp", "px")`** returns, depending on the device's resolution, for example, **`"22px"`** on one device and **`"24px"`** on another

# 15.2 How to Set Styles

This topic describes where you can set styles for the different components of a page design and how settings at these various locations interact with each other. Broadly, stye properties can be defined for pages and for each control that has been placed on the page. The Table control is special because it can be used for laying out components neatly on a page and because it has a hierarchical structure, each level of which has its own style properties (table, row, column, cell). The individual style properties are described in detail in the topics about page properties 384 and individual controls 403.

## The Styles & Properties Pane

All styles can be set in the context-sensitive Styles & Properties Pane 274 (*screenshot below*). When a control is selected in the design, the properties of that control are displayed at the top of the pane. In the example screenshot below, a Chart control 438 was selected in the design. To set a value for any style property of the chart, select from the available values or enter the value you want, either directly or via an XPath expression. The style properties of ancestor components are displayed below the selected component—with higher-level ancestors being displayed progressively lower in the pane. For example, we can deduce from the screenshot below, that the chart control has been placed in a cell of a top-level table of the page. So, with the chart control selected, we can set the style properties not only of the chart, but also of the cell, column, row, and table that contain the chart. (To select another column or row of the table, you will need to select a component in the respective column or row.) With the chart selected, you can also set style properties of the page such as its margins and background color.



Styles & Properties

| | |
|---|---|
| Control | |
| Control Kind | Chart |
| Name | Chart1 |
| Chart Settings | |
| ID | |
| Create Before Load | |
| Chart Creation Width | |
| Chart Creation Height | |
| Control Action | |
| Visible | |
| Tooltip | |
| Horizontal Alignment | |
| Vertical Alignment | |
| Control Width | |
| Max Control Width | |
| Control Height | |
| Limit Control Height to Ca... | |
| Margin | |
| On Enter/Escape | |
| Style Sheet | |
| Browser CSS Class | |
| Table Cell | |
| Table Column | |
| Table Row | |
| Table | |
| Page | |
| Name | SplashScreens |
| Show Page Title Bar | |
| Page Title | |
| Auto Add Submit Button | |
| Submit On Assertion | |
| Page Actions | |
| Audio Recording Actions | |
| Assertion | |
| Assertion Message | |
| Background Color | |
| Margin | |
| Style Sheet | |
| Browser Max Width | |
| Browser CSS Class | |
| Project | |

**Note:** When using tables, there can be some interaction between table components and what's in the table's cells. For example, a table column can be specified (with the `wrap_content` value of the column's *Width* property) to be only just as wide as the column's content (which could be an Image[541] or a Label[554]). For more information about using table layouts well, see the descriptions of the various table properties[614].

## Style sheets and the cascading of styles

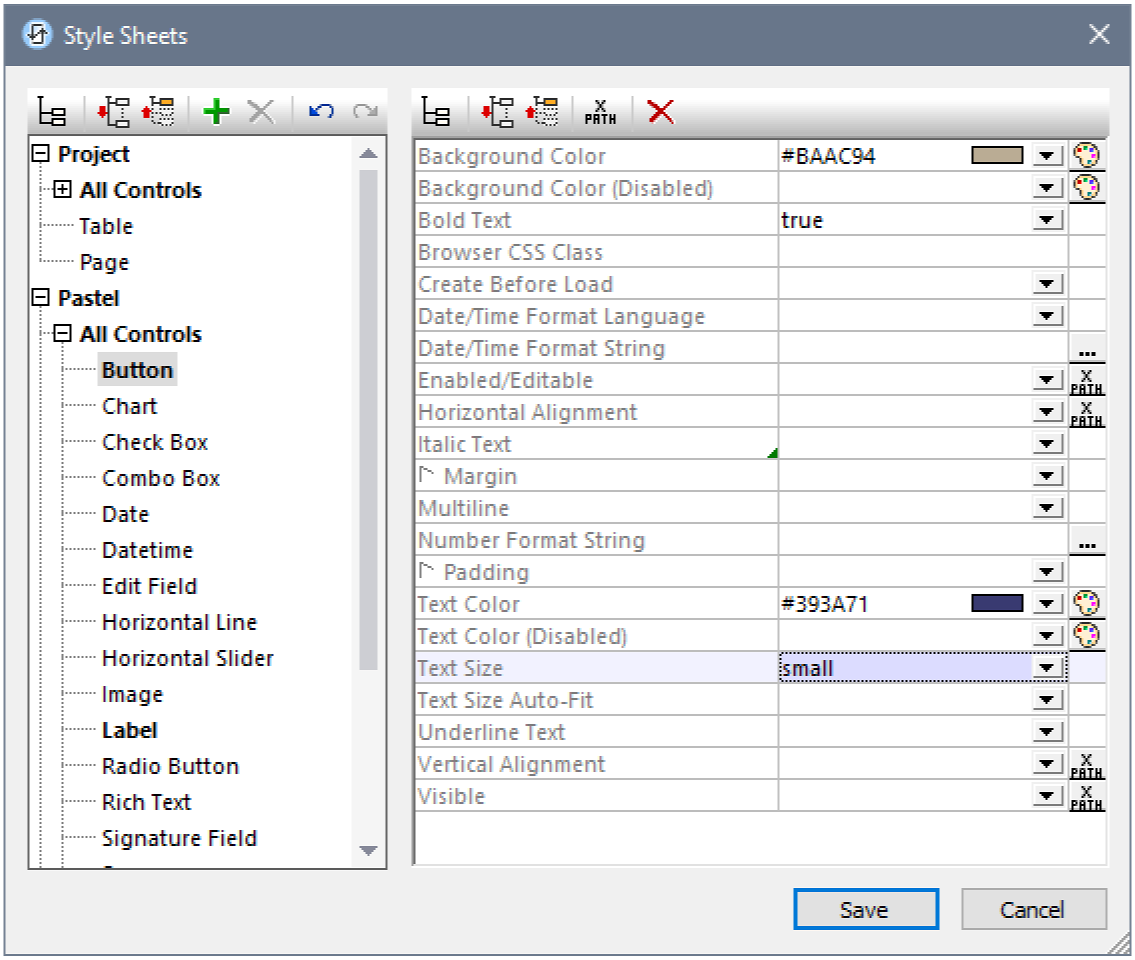
MobileTogether's Style Sheets[1318] feature enable you to use one or more style sheets to define style properties that can be used across all pages of the project. In the *Project* style sheet, you can define a style for each type of control (say, for all Buttons[417]), for all tables, and for all pages of the project. This enables you to define default styles for the entire project and then override them in specific instances. This cascading of styles works as follows:

- There is always one default *Project* style sheet. Any style you define here will be automatically used across the project.
- You can create any number of custom named style sheets. A custom style sheet overrides the *Project* style sheet. So, if on a control, you want to override the styles of the *Project* style sheet with those of a custom style sheet, then you select, on the control's *Style Sheet* property, the custom style sheet that you want to use. You can override the *Project* styles of tables and pages in the same way—that is, by selecting a custom style sheet for the respective table or page.
- If you select a component and assign it a style value directly in the Styles & Properties Pane, then this value has a higher priority than that of any custom style sheet that might been assigned to the component.

For a detailed description of style sheets[1318], go to the next section[1318].

# 15.3 Style Sheets

MobileTogether Designer's Style Sheets feature enables you to define global styles that can be applied at the project, page, table, and control level. Style sheets are created and defined in the Style Sheets dialog (*screenshot below*), which is accessed by clicking the **Project | Style Sheets** command. You can create multiple user-defined style sheets. These style sheets can then be applied to various components of the design 1327.



- Project
  - All Controls
  - Table
  - Page
- Pastel
  - All Controls
    - Button
    - Chart
    - Check Box
    - Combo Box
    - Date
    - Datetime
    - Edit Field
    - Horizontal Line
    - Horizontal Slider
    - Image
    - Label
    - Radio Button
    - Rich Text
    - Signature Field

| | |
|---|---|
| Background Color | #BAAC94 |
| Background Color (Disabled) | |
| Bold Text | true |
| Browser CSS Class | |
| Create Before Load | |
| Date/Time Format Language | |
| Date/Time Format String | |
| Enabled/Editable | |
| Horizontal Alignment | |
| Italic Text | |
| Margin | |
| Multiline | |
| Number Format String | |
| Padding | |
| Text Color | #393A71 |
| Text Color (Disabled) | |
| Text Size | small |
| Text Size Auto-Fit | |
| Underline Text | |
| Vertical Alignment | |
| Visible | |

## Adding, copying, and deleting user-created style sheets

There are two types of style sheets: (i) a Project style sheet which is applied **automatically** at the project level and cannot be deleted; (ii) user-created style sheets, which can be applied separately to individual pages, tables, and controls. *See Style Sheet Type and Scope 1320 for more information.*

⊟ *Icons in this section*

＋ **Add Style Sheet**

☒. **Delete Style Sheet**

- To add a user-created style sheet, click **Add Style Sheet** .
- To copy a project or user-defined style sheet, do the following: (i) select the style sheet's name (in the screenshot above, for example, the name of one style sheet is *Project* and the name of the other is *Pastel*), (ii) press **Ctrl+C** to copy to the clipboard, (iii) press **Ctrl+V** to paste as a new style sheet. Alternatively you can use the **Copy** and **Paste** commands of the style sheet's context menu. This way you can start a new style sheet so that it already contains the styles of a pre-existing style sheet; you can then add or modify styles in the new style sheet.
- To rename a user-created style sheet, double-click the style sheet's name and edit it.
- To delete a user-created style sheet, click **Delete Style Sheet**.

## Defining styles

In the left-hand pane, within a style sheet, select a level (page, table, or control) at which you wish to define a style, then, in the right hand pane, assign a value to that particular style property. You can select or enter a static property value, or you can enter an XPath expression that evaluates to a property value. An example of a dynamic assignment would be to make a property value conditional on some criterion, such as the screen width of the end-user mobile device.

## Priority of style definitions

The closer the location of a style definition is to a component, the higher will be that style definition's priority (*see Priority across Style Sheets*[1325]) relative to a definition for the same property at a location farther away. For example, if a user-created style sheet is applied to, say, a Button instance[417], then the styles in this user-created style sheet will have a higher priority (with regard to that button's style properties) than styles in the *Project* style sheet. In this way, you can provide design components with cascading styles. Furthermore, priority levels within a style sheet[1321] itself provide you with additional flexibility for defining cascading effects.

A user-created style sheet can be applied[1327] to a design component by entering the name of the style sheet as the value of that component's `Style Sheet` property. The style sheet assignment can be made statically (by entering the name directly) or dynamically (via an XPath expression). Being able to use XPath expressions enables you to select user-created style sheets according to the dynamic context. For example, you can make the choice of style sheet dependent on the type of the current end-user mobile device.

## About the Project CSS File

There is a MobileTogether Designer feature that involves styling for web clients—that is, browsers—only, but which is separate from the Style Sheets feature. This is the Project CSS File feature, which assigns a CSS file to a project by means of the project's Browser Settings[296]. In a Project CSS File, you can define styles for classes that are assigned to design components via each component's `Browser CSS Class` property. In the current section, we will **not** be dealing with the Project CSS File. For information about this feature, see the description of the project's Browser Settings[296].

## This section

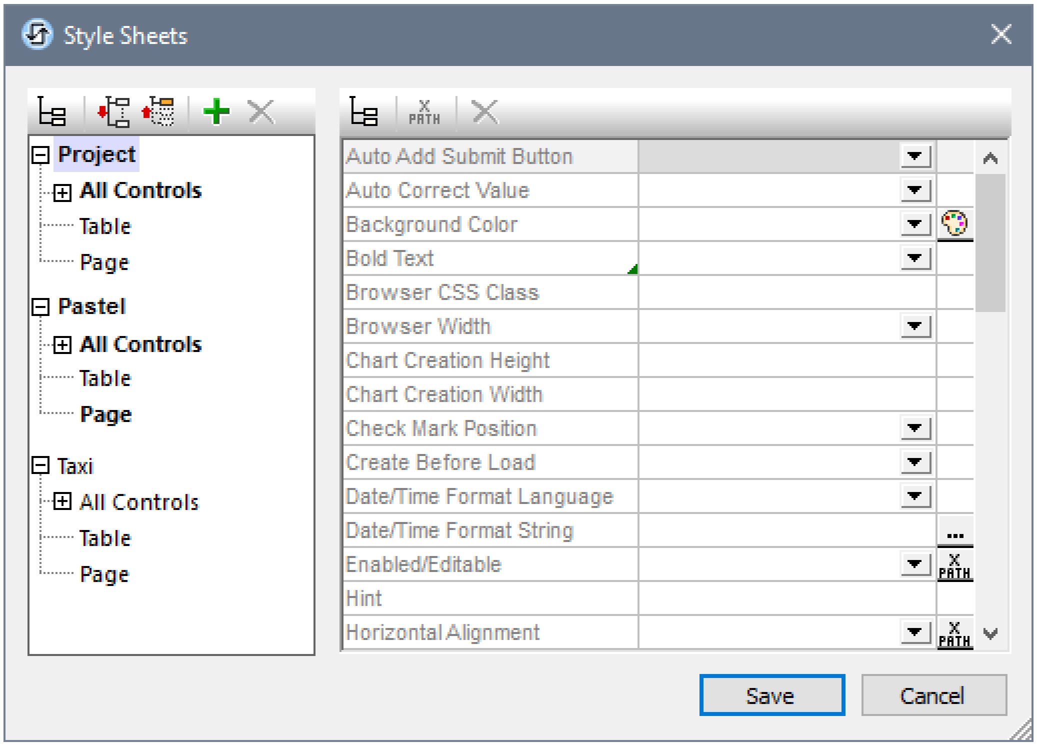
This section is organized as follows:

- Style Sheet Type and Scope[1320] describes the two types of style sheet and their respective scopes
- Priority within a Style Sheet[1321] describes the priority levels available within a single style sheet

- Priority across Style Sheets[1325] describes how styles can be prioritized when multiple style sheets are used
- Applying User-Created Style Sheets[1327] shows how user-created style sheets can be applied to design components
- Style Sheet Properties[1328] provides an overview of how to work with component styles in the Style Sheets dialog

## 15.3.1 Style Sheet Type and Scope

On the basis of their scope, style sheets can be grouped into two types:



- A project style sheet named *Project*; this name is fixed and cannot be changed. The styles defined in the *Project* style sheet are **applied automatically** across the **entire project** (that is, across all the pages of the project). *Project* styles can be overridden by styles defined at a location with a higher priority.
- User-created style sheets, each with a user-given name. You can create any number of these style sheets. (In the screenshot above, there are two user-created style sheets: *Pastel* and *Taxi*.) The styles defined in a user-created style sheet can be applied to individual controls, tables, and/or pages. This is done by entering the name of the user-created style sheet as the value of the `Style Sheet` property of the respective control, table, or page. When a user-created style sheet is applied in this way, it will have a higher priority than the *Project* style sheet.

**Note:** To add a user-created style sheet, click **Add Style Sheet** +. To rename a user-created style sheet, double-click the name and edit.

**Note:** If a style group (*Controls, Table,* or *Page*) contains at least one style definition, then that style group and its containing style sheet are displayed in bold; otherwise these are displayed in normal fontface. For example, in the screenshot above, no style has been defined in the *Taxi* style sheet, but at least one style has been defined in the other two style sheets. In the *Project* style sheet, at least one style has been defined for at least one control. The *Pastel* style sheet has at least one style defined in the *Page* group of styles and for at least one control.

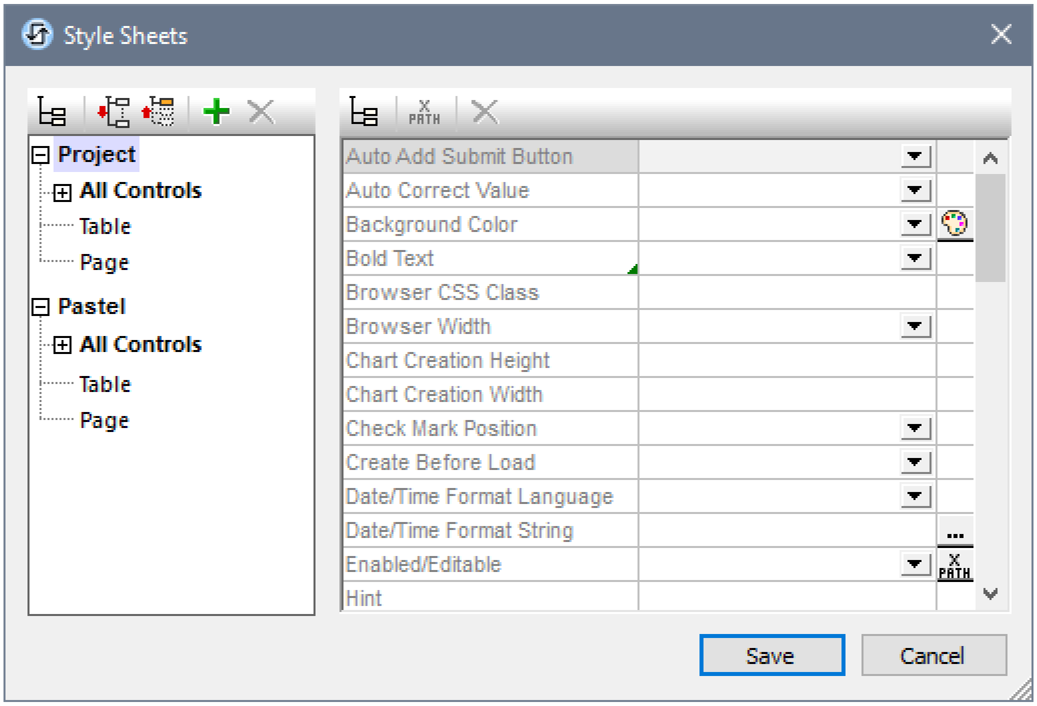
**Note:** Each pane of the dialog (left and right) has an icon that enables/disables the display of non-empty items. Displaying only the non-empty items is useful when you wish to see a list of only the styles that have been defined; for example, when you want an overview of currently defined styles. The left-hand pane also has toolbar icons for (i) expanding all items, and (ii) collapsing all items.

## 15.3.2 Priority within a Style Sheet

Both the *Project* style sheet and user-created style sheets are structured into three levels:

```
Style Sheet (Level-1)
|
|-- All Controls (Level-2)
|   |
|   |-- ControlType-1 (Level-3)
|   |   ...
|   |-- ControlType-n (Level-3)
|
|-- Table (Level-2)
|
|-- Page (Level-2)
```

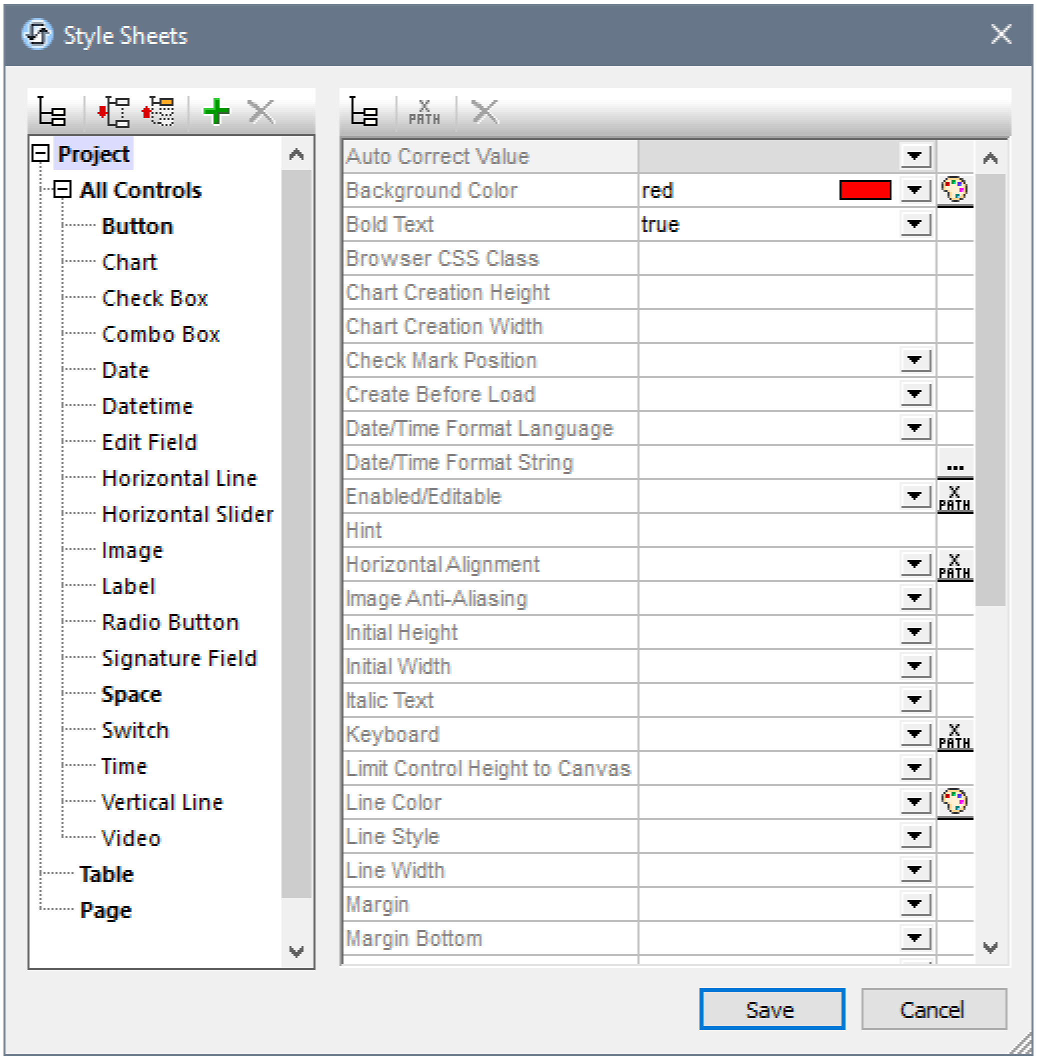
You can also see this hierarchy in the screenshots below.



Each higher level in the hierarchy passes all its style properties to the level below it. So the *All Controls* level passes all its style definitions to the control types on Level 3. The principle is that, if a property value is set at the *All Controls* level, then all the controls (on the lower level) that have this property will inherit the property value that was set at the *All Controls* level.

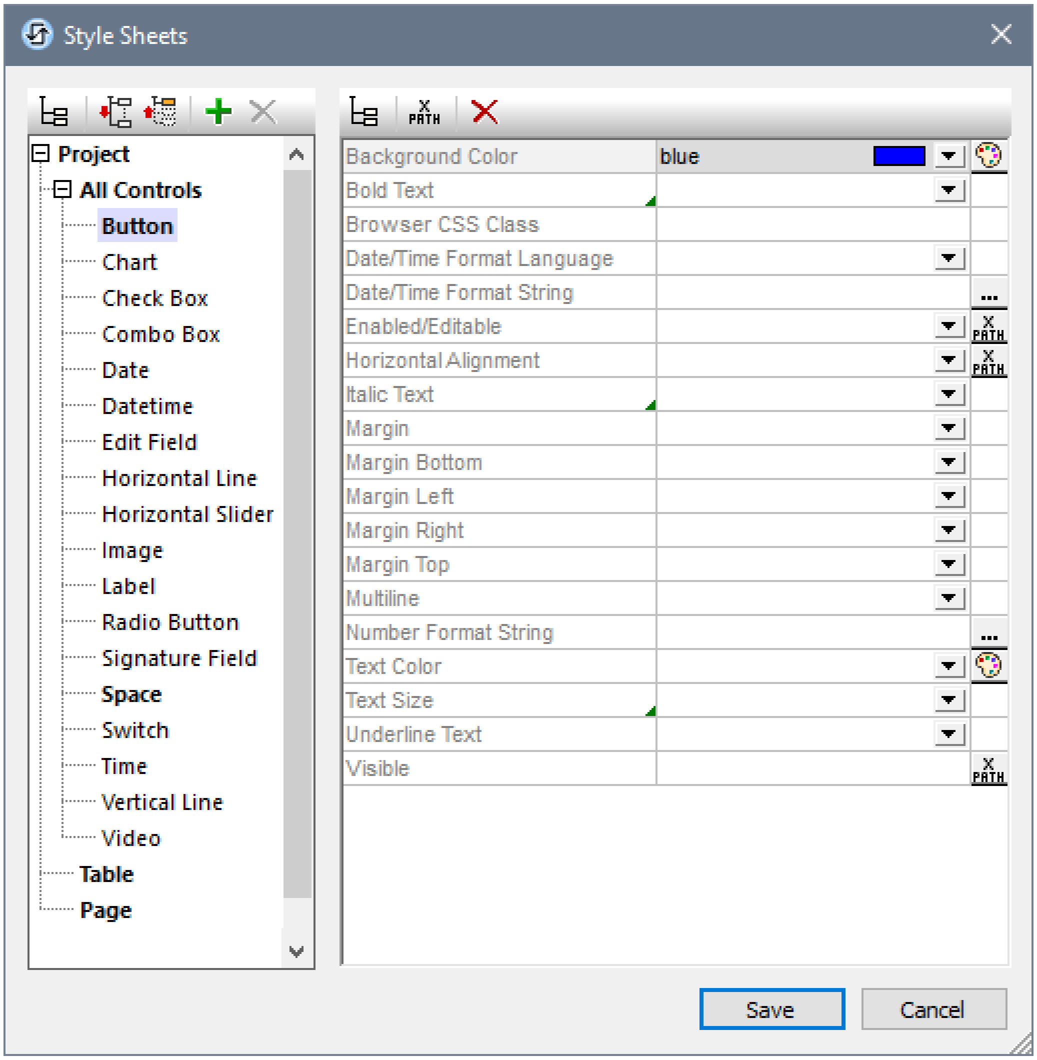
The properties that are available at the *Style Sheet* level are all the properties of the *All Controls* level—plus the *Table* and *Page* properties (which define, respectively, properties for tables and pages). So, for example, if you set the **Background Color** property on the *Style Sheet* level (say a value of **red**, as in the screenshot below left), then all the control types of Level 3 that have a **Background Color** property, plus any tables in the design, as well as the design page itself will all inherit this value (**red** in this case)—as long as no definition exists for these descendant properties.

If you wish to override a property value that was assigned at a higher level, then assign an overriding value at the lower level. In the screenshot below right, for example, the Button control type has been assigned a **Background Color** property value of **blue**. So while all control types that have a **Background Color** property (as well as tables and the page) will inherit a background color of **red** (from the higher-level assignment in the screenshot at left), all Button controls will have a background color of **blue**. If you wish to give one specific Button instance a background color other than **blue**, specify the color you want in that particular Button instance's **Background Color** property. (Do this by selecting the Button control in the design and setting its **Background Color** property value in the Styles and Properties Pane[274].)



- Project
  - All Controls
    - Button
    - Chart
    - Check Box
    - Combo Box
    - Date
    - Datetime
    - Edit Field
    - Horizontal Line
    - Horizontal Slider
    - Image
    - Label
    - Radio Button
    - Signature Field
    - Space
    - Switch
    - Time
    - Vertical Line
    - Video
  - Table
  - Page

| | |
|---|---|
| Auto Correct Value | |
| Background Color | red 🟥 |
| Bold Text | true |
| Browser CSS Class | |
| Chart Creation Height | |
| Chart Creation Width | |
| Check Mark Position | |
| Create Before Load | |
| Date/Time Format Language | |
| Date/Time Format String | |
| Enabled/Editable | |
| Hint | |
| Horizontal Alignment | |
| Image Anti-Aliasing | |
| Initial Height | |
| Initial Width | |
| Italic Text | |
| Keyboard | |
| Limit Control Height to Canvas | |
| Line Color | |
| Line Style | |
| Line Width | |
| Margin | |
| Margin Bottom | |



| | |
|---|---|
| Background Color | blue |
| Bold Text | |
| Browser CSS Class | |
| Date/Time Format Language | |
| Date/Time Format String | |
| Enabled/Editable | |
| Horizontal Alignment | |
| Italic Text | |
| Margin | |
| Margin Bottom | |
| Margin Left | |
| Margin Right | |
| Margin Top | |
| Multiline | |
| Number Format String | |
| Text Color | |
| Text Size | |
| Underline Text | |
| Visible | |

## Higher priority for definitions located closer to the design component

If a style property exists at multiple levels, then the definition that is relatively more specific to the design component has relatively higher priority. For example, a style sheet property definition on a control type has higher priority than a definition for the same property on the style sheet level.

The table below gives, for each column, the relative priority levels of the same style property if the property is set at multiple levels. Levels lower in the column have relatively higher priority. For example, in the first column, if a property style (say `background color`) is set on an individual control type (say Buttons), then the value of this style property will have a higher priority than a value set for the same style property at the *All controls* level or the *Style sheet* level.

| Style sheet property of control set on... | Style sheet property of table set on... | Style sheet property of page set on... |
|---|---|---|

| | | |
|---|---|---|
| `Style sheet (Level-1)` | `Style sheet (Level-1)` | `Style sheet (Level-1)` |
| `All controls (Level-2)` | `Table (Level-2)` | `Page (Level-2)` |
| `Individual control type (Level-3)` | | |

**Note:** To set a property for a single instance (rather than all instances) of an individual control type, or table, or page, select that instance in the design and assign it its own property value in the Styles and Properties Pane[274]. This definition will have a higher priority than a definition in a style sheet because it is specific to that design component, that is, on the design component directly.

### Style sheet: scope and application

The *Project* style sheet is applied automatically to the entire project. This means, for example, that a **`Background Color`** property value that is defined at the style sheet level of the *Project* style sheet will automatically be inherited by all the **`Background Color`** properties in the project.

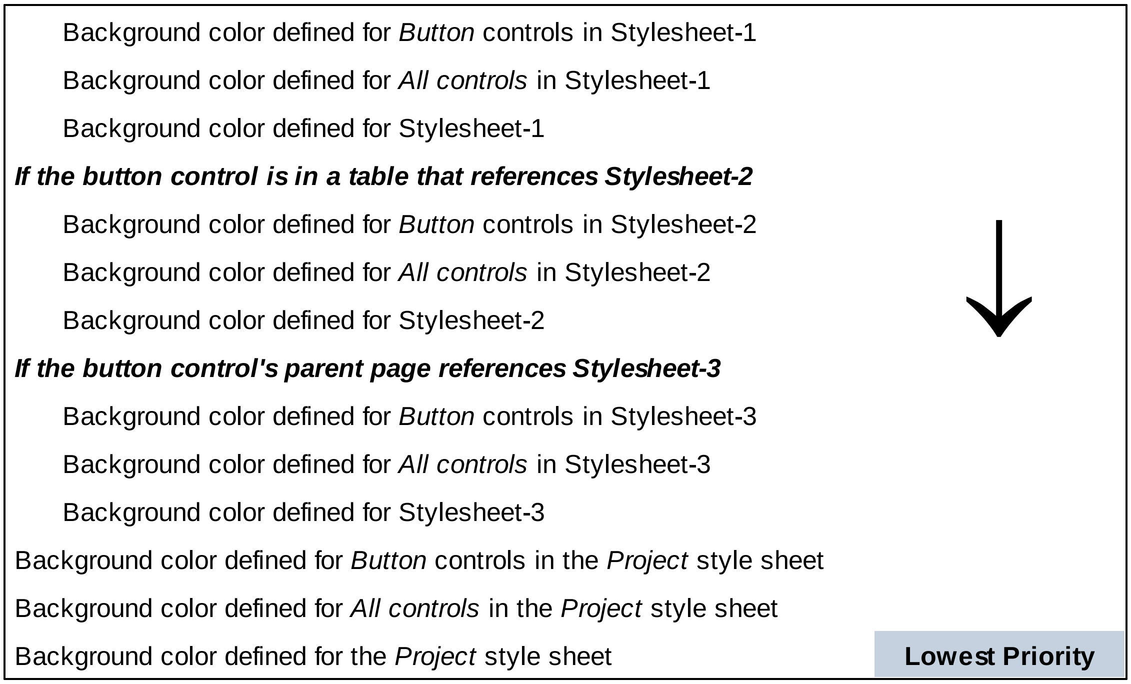
A user-created style sheet, on the other hand, can be applied only to instances of pages, tables, and individual controls; it cannot be applied to the entire project. The table below shows which design components inherit the styles defined at a specific style sheet level when applied to a page, table, or control instance.

| Definition level in style sheet | When style sheet is set on page/table/control instance, style sheet applies to... | | |
|---|---|---|---|
| | **Page instance** | **Table instance** | **Control instance** |
| *Style sheet* | Page instance; all tables and all controls on page | Table instance; all controls in table | Control instance |
| *All controls* | All controls on page | All controls in table | Control instance |
| *Control type* | All controls of that type on page | All controls of that type in table | Control instance if of that type |
| *Table* | All tables on page | Table instance | -- |
| *Page* | Page instance | -- | -- |

## 15.3.3 Priority across Style Sheets

The question of priority also arises when there are multiple definitions (at different levels, and in different style sheets) for a single style property, and when more than one of these definitions applies to the style property of a single design component. In this case, MobileTogether will look through definitions for this property at the various style sheet levels in the order given below. The first one to match is the one that is used. The table below uses the example of the **`background color`** property on a Button control.

| | |
|---|---|
| Background color defined on the button control in the design | **Highest Priority** |
| ***If the button control in the design references Stylesheet-1*** | |



Background color defined for *Button* controls in Stylesheet-1

Background color defined for *All controls* in Stylesheet-1

Background color defined for Stylesheet-1

***If the button control is in a table that references Stylesheet-2***

Background color defined for *Button* controls in Stylesheet-2

Background color defined for *All controls* in Stylesheet-2

Background color defined for Stylesheet-2

***If the button control's parent page references Stylesheet-3***

Background color defined for *Button* controls in Stylesheet-3

Background color defined for *All controls* in Stylesheet-3

Background color defined for Stylesheet-3

Background color defined for *Button* controls in the *Project* style sheet

Background color defined for *All controls* in the *Project* style sheet

Background color defined for the *Project* style sheet

**Lowest Priority**

If the property value is defined via an XPath expression, note the following:

- If the expression evaluates to an empty sequence, then the list is searched from top to bottom.
- If the expression is defined for a property that takes a Boolean value (such as the properties `visibility`, `bold`, and `italic`), any return value that is not **`true`** is, according to the rules of XPath, a value of **`false`**. As a result, the list is not searched further.

## Platform default values

Each mobile device platform (Android, iOS, Windows) has default values for certain style properties. For example, the default page background color of an iOS device might be white, whereas that of an Android device might be black. Note, however, that platform defaults are not available for all properties. You can use the **Set Platform Default Value** command to set a property value to the platform default of that property. Platform default values can be set at the following definitions levels:

- Directly on a design component: Right-click a design component's property definition in the Styles and Properties Pane, and select **Set Platform Default Value**.
- On a property in a style sheet: Right-click a property that is defined at any style sheet level (Project, All Controls, specific control type, table, and page), and select **Set Platform Default Value**.

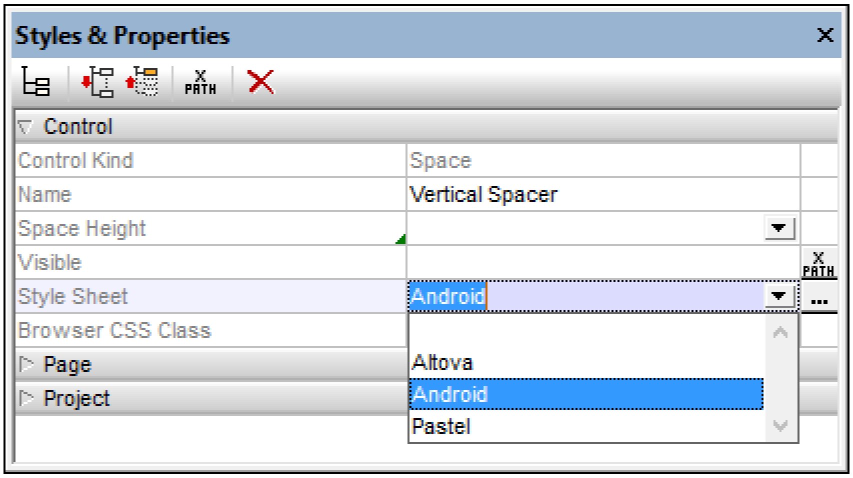
Like any other style definition, platform default values can override values that are defined relatively farther away (from the component), and can be overridden by style definitions that are relatively closer to the component.

## 15.3.4 Applying User-Created Style Sheets

A user-created style sheet can be applied to page instances, table instances, and control instances. The style definitions in the user-created style sheet will be immediately applied to the selected design component and will override existing style definitions of a lower priority.

You can apply a user-created style sheet to a design component (page, table, or control) as follows:

1. In the design, select the design component (page, table, or control) to which you wish to apply a user-created style sheet.
2. In the Styles & Properties Pane, select the **`Style Sheet`** property of the page, table, or control to which you wish to apply the style sheet. In the screenshot below, the **`Style Sheet`** property of a control has been selected.



| | |
|---|---|
| Control Kind | Space |
| Name | Vertical Spacer |
| Space Height | |
| Visible | |
| Style Sheet | Android |
| Browser CSS Class | |

3. In the dropdown list of the Style Sheet property's combo box (*see screenshot above*), select the user-created style sheet you wish to apply to the design component. (The dropdown list contains the names of all the user-created style sheets of the current project.) Alternatively, click the XPath icon in the pane's toolbar, and enter an XPath expression that will evaluate to the name of the style sheet you wish to apply.

**Note:** If a design component has a style assigned to it via a style sheet, then this is indicated by a green marker at the bottom right of the cell containing the property's name (*see the* **`Space Height`** *property in the screenshot above*). Placing the mouse cursor over the marker causes the style sheet information to be displayed in a pop up. Clicking the marker, takes you to the corresponding definition in the Style Sheets dialog 1318.

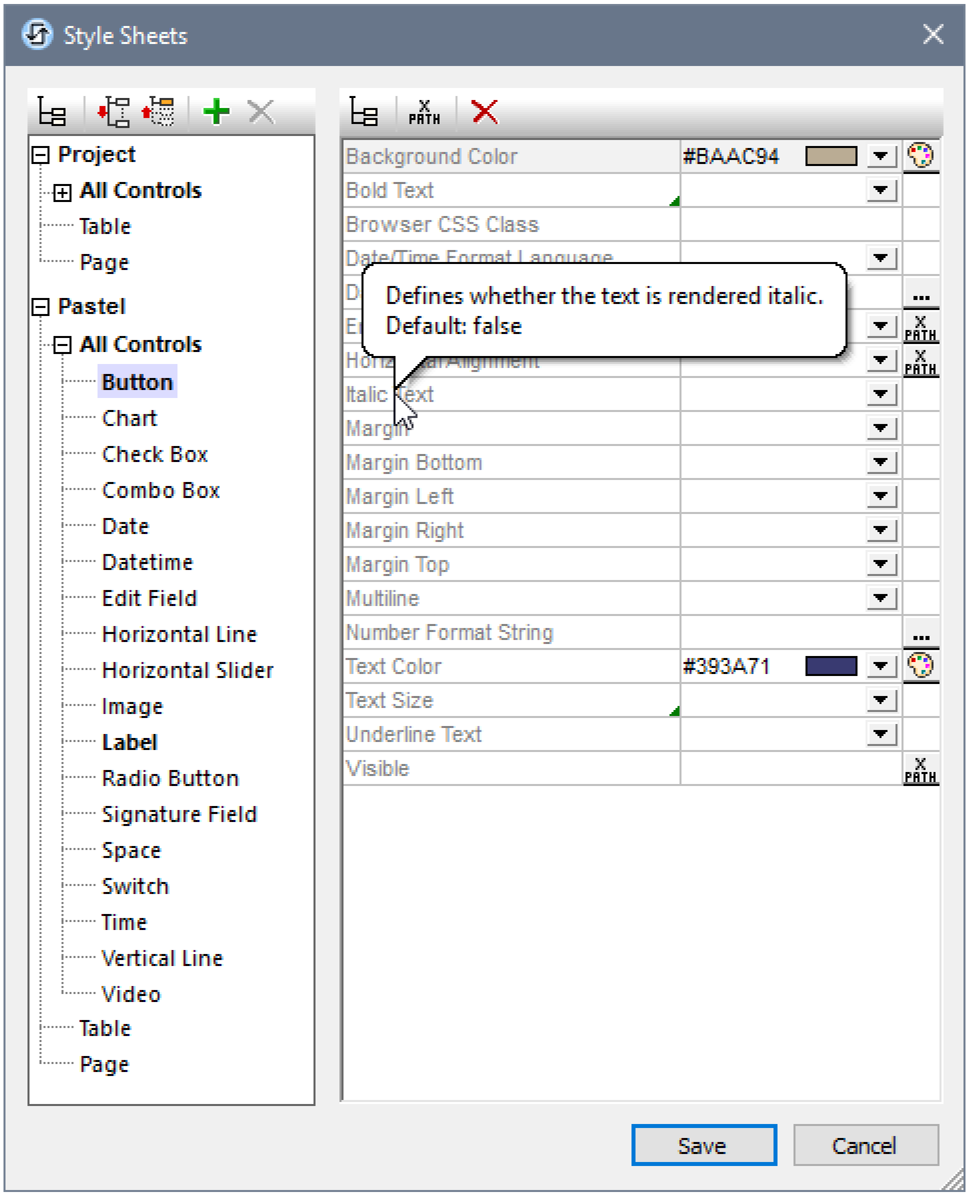
### Advantages of a style sheet selection via XPath

A big advantage of using an XPath expression to select a user-created style sheet is that the selection can be made conditional upon dynamic environmental criteria. For example, if you wish to specify one style sheet for iOS devices and another for all other devices, you could use the following XPath expression: `if ($MT_iOS=true()) then 'iOSStyleSheet' else 'GeneralStyleSheet'`.

**Note:** Switching style sheets often at runtime could cause solution execution to slow down.

## 15.3.5 Style Sheet Properties

In a style sheet, you can define styles for individual control types (such as Button controls and Label controls), tables, and pages. Select the design component for which you wish to define styles in the left-hand pane. The screenshot below shows that the Button control type has been selected. The properties of the selected design component appear in the right-hand pane. You can now select or enter values for individual properties. Click **Save** when done.



| | |
|---|---|
| Background Color | #BAAC94 |
| Bold Text | |
| Browser CSS Class | |
| Date/Time Format Language | |
| D | |
| E | |
| Hori[illegible]al Alignment | |
| Italic Text | |
| Margin | |
| Margin Bottom | |
| Margin Left | |
| Margin Right | |
| Margin Top | |
| Multiline | |
| Number Format String | |
| Text Color | #393A71 |
| Text Size | |
| Underline Text | |
| Visible | |

Defines whether the text is rendered italic.
Default: false

Note the following points:

- If you hover your mouse cursor over the name of a property, then a popup displays information about that property, including the property's default value (*see screenshot*).
- For properties that take a color value, click the property's color picker to quickly select a suitable color.

- You can enter an XPath expression as the value of a property. Do this by clicking either the XPath icon on the right-hand side of the property field (if available) or the **XPath** icon in the dialog's toolbar.
- Click the toolbar icon *List Non-Empty* to show only those properties that are not empty. This enables you to eliminate clutter and, consequently, to see only the currently defined styles of the selected design component.
- To remove a property value that you have set, click the **Reset** icon in the dialog's toolbar.
- You can also set the values of some properties to be the platform default value 1325 of that property. Do this by right-clicking the property and selecting **Set Platform Default Value**.

## Listing and grouping of properties

The context menu of a property (obtained by right-clicking the property) contains commands to list or group controls that are related to that property in the following ways:

- *List controls with the same direct style value:* Searches the design for the same property *value* as that of the currently selected property, and lists all controls that have this property value *set at the control level*. For example, if a label and a button have their `Text Color` property set to `red`, then both controls are listed when the command is executed for this property value.
- *List controls with the same style value:* Searches the design for the same property *value* as that of the currently selected property, and lists all controls that have this property value—whether the value is set at the control level or via a style sheet. For example, if a label and a button have their `Text Color` property set to `red`, and a style sheet containing a `Text Color` style of `red` is applied to a combo box, then all three controls will be listed when this command is executed for this property value.
- *Group controls by direct style value:* All controls in the design are grouped on the values of the currently selected *property* if these values are *set on the control itself*. For example, if the currently selected *property* is `Text Color`, then all controls that have the same `Text Color` value, set at the control level, are grouped together. So all controls that have, say, a `Text Color` value of `red` set at the control level are listed together, while those with another value are listed in their own group.
- *Group controls by style value:* All controls in the design are grouped on the values of the currently selected *property*—whether these values are set at the control level or via a style sheet.. For example, if the currently selected *property* is `Text Color`, then all controls that have the same `Text Color` value, set at the control level or via a style sheet, are grouped together. So all controls that have, say, a `Text Color` value of `red` are listed together, while those with another value are listed in their own group.

The lists are displayed in the Listings Pane 281, and controls in these lists are hyperlinked

# 15.4 Style Variance Across Clients

Some MobileTogether solutions might look different across operating systems (platforms) because each operating system handles a few styling properties differently. (The different platforms on which MobileTogether solutions run are: Android, iOS, Windows, and Web.)

This topic:

- explains how you can create a uniform look across platforms, and
- lists the style properties that are handled differently by specific platforms.

## Create a uniform look across platforms

The most effective way to set a uniform look is to identify style properties that are handled differently across platforms and then set the value of each such property via an XPath expression that sets different values for different platforms. You can build conditional branches in the XPath expression by using the static global variables that hold the platform information[1300].

For example, the following XPath expression can be set on the *Padding* style property of project components:

```
if ($MT_Android=true()) then "0dp" else "1dp"
```

The XPath expression above sets one padding value for Android devices and another for all other platforms.

You can set these style property definitions at one or more of the following locations:

- *Via the design layout:* On individual controls; on the page; in the project settings
- *Via style sheets:* On all controls together; on a specific control type; on the tables of a page; on a page

In the project's properties settings[296], you can set *UI Compatibility Mode* to `true` to set default values of properties to be the same across platforms.

## Platform-specific handling of particular styles

Given below is a list of style properties that are handled in a noticeably different way by at least one platform.

*General*
The default colors, fonts, and sizes vary across platforms and also across different devices on a single platform, Additionally, controls look different across devices and across different versions of a single operating system.

*Margins of top-level controls*
Top-level controls are controls that are located directly within the page container. Put another way, these are controls that are not inside a table. Android devices set a default margin of **`9px`** for all top-level controls (although the Label control[554] has a bottom margin of `0px`). Other platforms have other defaults. You can use the Top-Level Margins property of the More Project Settings dialog[296] to set the margins of top-level controls uniformly across all platforms.

*Label controls*
On Android, the Label control[554] has a default margin of **`9px`** on all sides except the bottom, which has a margin of **`0px`**.

*Padding*

- On Windows, all controls (except the Label control 554) have a "natural padding" of **`1px`**. This is overridden when you set padding on a control (for example, **`0px`**).
- On iOS, a standard table 614 padding is applied: **`9px`** on right and left, and **`5px`** on top and bottom. If values are set for any *Padding* property of individual tables, then that *Padding* value (top, right, bottom, or left) is added to the respective iOS table padding value. If you wish to remove the standard iOS table padding, set the *iOS Table Padding* 296 property value (in a project's *More Project Settings* dialog) to **`false`**.

*Buttons*

- Android buttons have a “natural padding”—even if `0px` is specified.
- iOS buttons have no background fill. Instead, they have a "tint color".
- Windows dark theme buttons are transparent.

*Back Button*

On iOS, there is no device-supplied **Back** button that is always available. You will need to explicitly add an option (button or an alternative) that enables iOS users to exit the solution.

*Combo Box controls*

On iOS, multiline combo boxes 459 are not supported.

*Switch controls*

On iOS, Switch controls 603 have no text.

*DateTime controls*

The DateTime control 484 is available on iOS only.

*Rich Text controls*

On Windows clients, the Rich Text control 584 can be edited on a PC (in a web client), but cannot be edited on Windows Phone.

*Browser Max Width, Browser CSS Class*

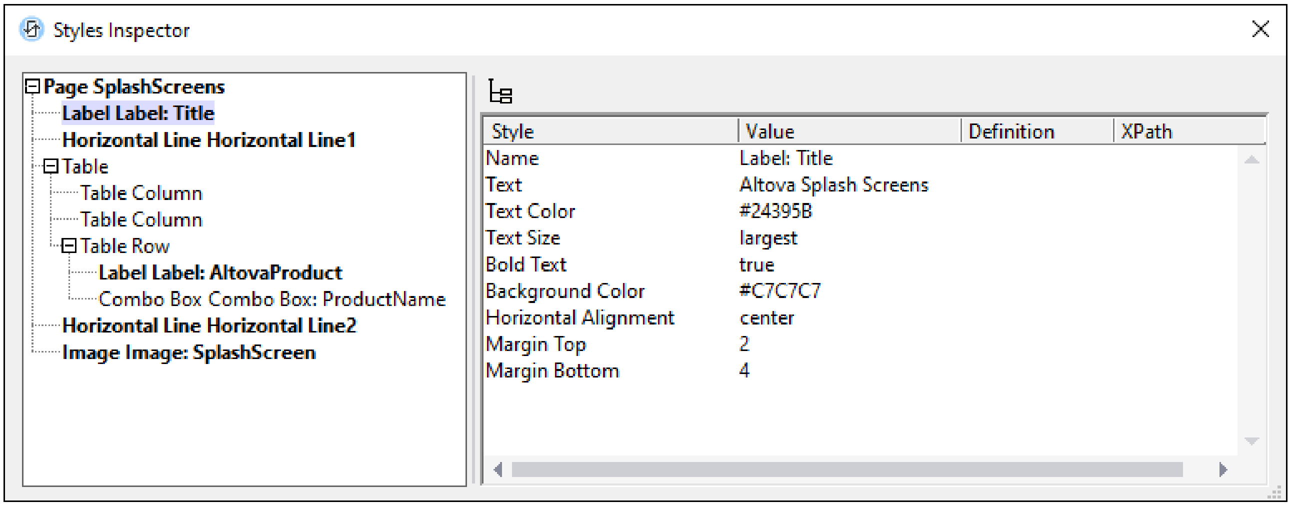
These page properties 384 apply to Web browser display only.

*Project's Browser settings*

The project's browser settings 296 apply to Web browser display only.

# 15.5 Styles Inspector

Styles Inspector (*screenshot below*) can be opened during simulations to provide an overview of the computed styles of the controls of the current page.



- **Page SplashScreens**
  - **Label Label: Title**
  - **Horizontal Line Horizontal Line1**
  - Table
    - Table Column
    - Table Column
    - Table Row
      - **Label Label: AltovaProduct**
      - Combo Box Combo Box: ProductName
  - **Horizontal Line Horizontal Line2**
  - **Image Image: SplashScreen**

| Style | Value | Definition | XPath |
|---|---|---|---|
| Name | Label: Title | | |
| Text | Altova Splash Screens | | |
| Text Color | #24395B | | |
| Text Size | largest | | |
| Bold Text | true | | |
| Background Color | #C7C7C7 | | |
| Horizontal Alignment | center | | |
| Margin Top | 2 | | |
| Margin Bottom | 4 | | |

## Open Styles Inspector

Styles Inspector can be opened during simulations 1355 in the following ways:

- Click the Styles Inspector toolbar button.
- Ctrl-Click a control (which auto-selects that control in Styles Inspector).

## Description

Styles Inspector consists of two areas: (i) The left pane shows a document tree of the controls of the current page; (ii) the right pane shows the computed styles of the control selected in the left pane.

*Left pane*
The left pane shows the controls of the page in a hierarchical structure.

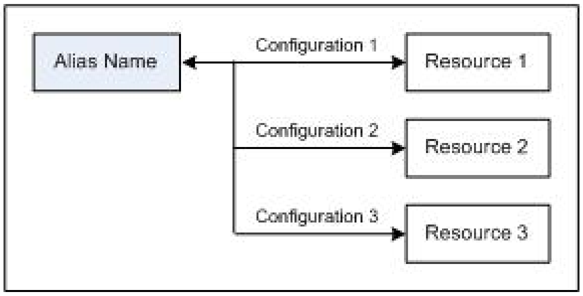
- If a control is in bold font, then it has at least one style with a non-default value.
- If a control is in normal font, then it has no style that has a non-default value.
- If a control is in light gray, then the control is invisible.
- On clicking a control: (i) the control flashes red in the simulator, (ii) the controls properties are displayed in the Styles & Properties Pane 274, and (iii) the control's styles are displayed in the right pane of Styles Inspector.
- When Styles Inspector is open and you Ctrl-Click a control in the simulator, then that control will be selected in Styles Inspector.

*Right pane*
The right pane shows the computed styles of the control selected in the left pane.

- The **Hide Default Styles** icon at the top left of the pane toggles the display of the control's styles between showing (i) only styles with non-default values and (ii) all styles (including those with default values).
- When a style is clicked, its definition in the Styles & Properties Pane[274] is selected.
- The pane has columns for: (i) the name of the style, (ii) the value of the style, and (iii) the origin of the value.

# 16 Altova Global Resources

Altova Global Resources is a collection of aliases for file, folder, and database resources. Each alias can have multiple configurations, and each configuration maps to a single resource (*see screenshot below*). Therefore, when a global resource is used as an input, the global resource can be switched among its configurations. This is done easily via controls in the GUI that let you select the active configuration. For example, you can specify a global resource as the default file of a page data source and switch resources by switching the active configuration in the GUI.



Using Altova Global Resources involves two processes:

- Defining Global Resources [1335]: Resources are defined and the definitions are stored in an XML file. These resources can be shared across multiple Altova applications.
- Using Global Resources [1344]: Within MobileTogether Designer, files can be located via a global resource instead of via a file path. The advantage is that the resource can be switched by changing the active configuration in MobileTogether Designer.

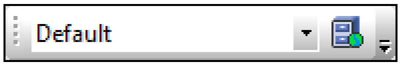
## Global resources in other Altova products

Currently, global resources can be defined and used in the following individual Altova products: XMLSpy, StyleVision, MapForce, Authentic Desktop, MobileTogether Designer, and DatabaseSpy.

# 16.1 Defining Global Resources

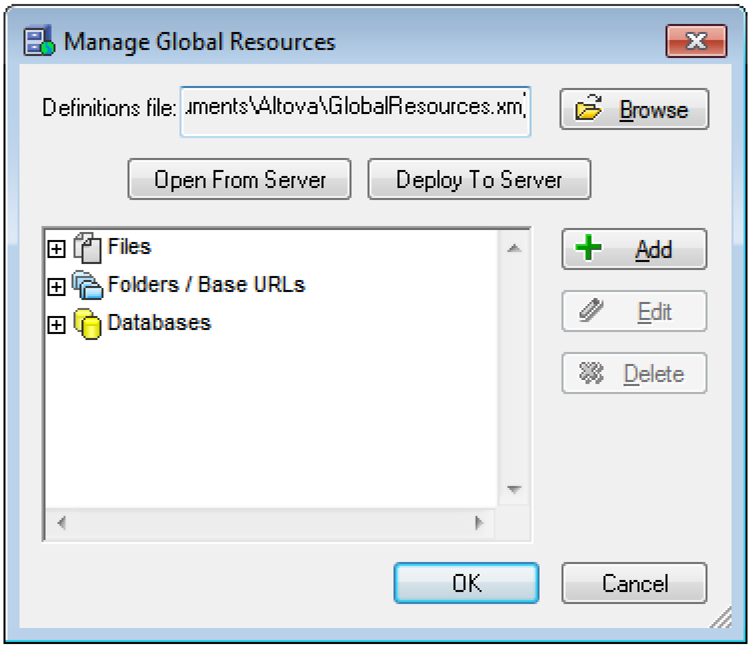
Altova Global Resources are defined in the Manage Global Resources dialog, which can be accessed in two ways:

- Click the menu command **Tools | Global Resources**.
- Click the **Manage Global Resources** icon in the Global Resources toolbar (*screenshot below*).



## The Global Resources Definitions file

Information about global resources is stored in an XML file called the Global Resources Definitions file. This file is created when the first global resource is defined in the Manage Global Resources dialog (*screenshot below*) and saved.



Definitions file: ıments\Altova\GlobalResources.xml

When you open the Manage Global Resources dialog for the first time, the default location and name of the Global Resources Definitions file is specified in the *Definitions File* text box (*see screenshot above*):

```
C:\Users\<username>\My Documents\Altova\GlobalResources.xml
```

This file is set as the default Global Resources Definitions file for all Altova applications. So a global resource can be saved from any Altova application to this file and will be immediately available to all other Altova applications as a global resource. To define and save a global resource to the Global Resources Definitions file, add the global resource in the Manage Global Resources dialog and click **OK** to save.

To select an already existing Global Resources Definitions file to be the active definitions file of a particular Altova application, browse for it via the **Browse** button of the *Definitions File* text box (*see screenshot above*).

**Note:** You can name the Global Resources Definitions file anything you like and save it to any location accessible to your Altova applications. All you need to do in each application, is specify this file as the Global Resources Definitions file for that application (in the *Definitions File* text box). The resources become global across Altova products when you use a single definitions file across all Altova products.

**Note:** You can also create multiple Global Resources Definitions files. However, only one of these can be active at any time in a given Altova application, and only the definitions contained in this file will be available to the application. The availability of resources can therefore be restricted or made to overlap across products as required.

## Managing global resources: adding, editing, deleting, saving

In the Manage Global Resources dialog (*screenshot above*), you can add a global resource to the selected Global Resources Definitions file, or edit or delete a selected global resource. The Global Resources Definitions file organizes the global resources you add into groups: of files, folders, and databases (*see screenshot above*).

To ***add a global resource***, click the **Add** button and define the global resource in the appropriate Global Resource dialog that pops up (*see the descriptions of files[1337], folders[1340], and databases[1335] in the sub-sections of this section*). After you define a global resource and save it (by clicking **OK** in the Manage Global Resources dialog), the global resource is added to the library of global definitions in the selected Global Resources Definitions file. The global resource will be identified by an alias.

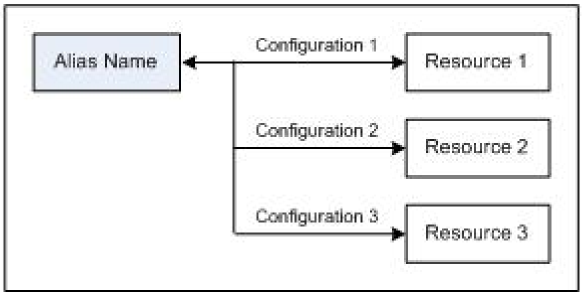
To ***edit a global resource***, select it and click **Edit**. This pops up the relevant Global Resource dialog, in which you can make the necessary changes (*see the descriptions of files[1337], folders[1340], and databases[1341] in the sub-sections of this section*).

To ***delete a global resource***, select it and click **Delete**.

After you finish adding, editing, or deleting, make sure to click **OK** in the Manage Global Resources dialog to ***save your modifications*** to the Global Resources Definitions file.

## Relating global resources to alias names via configurations

Defining a global resource involves mapping an alias name to a resource (file, folder, or database). A single alias name can be mapped to multiple resources. Each mapping is called a configuration. A single alias name can therefore be associated with several resources via different configurations (*screenshot below*).



In an Altova application, you can then assign aliases instead of files. For each alias you can switch between the resources mapped to that alias simply by changing the application's active Global Resource configuration (active configuration). For example, in MobileTogether Designer, if you have a data source with two or more alternative default files, you can assign a global resource alias as the default file. In MobileTogether Designer, you can then change the active configuration to use different default files. If `Configuration-1` maps `FirstDefault.xml` to the global resource alias, and `Configuration-1` is selected as the active configuration, then `FirstDefault.xml` will be used as the default file. In this way multiple configurations can be used to access multiple resources via a single alias. This mechanism can be useful when testing and comparing resources. Furthermore, since global resources can be used across Altova products, resources can be tested and compared across multiple Altova products as well.

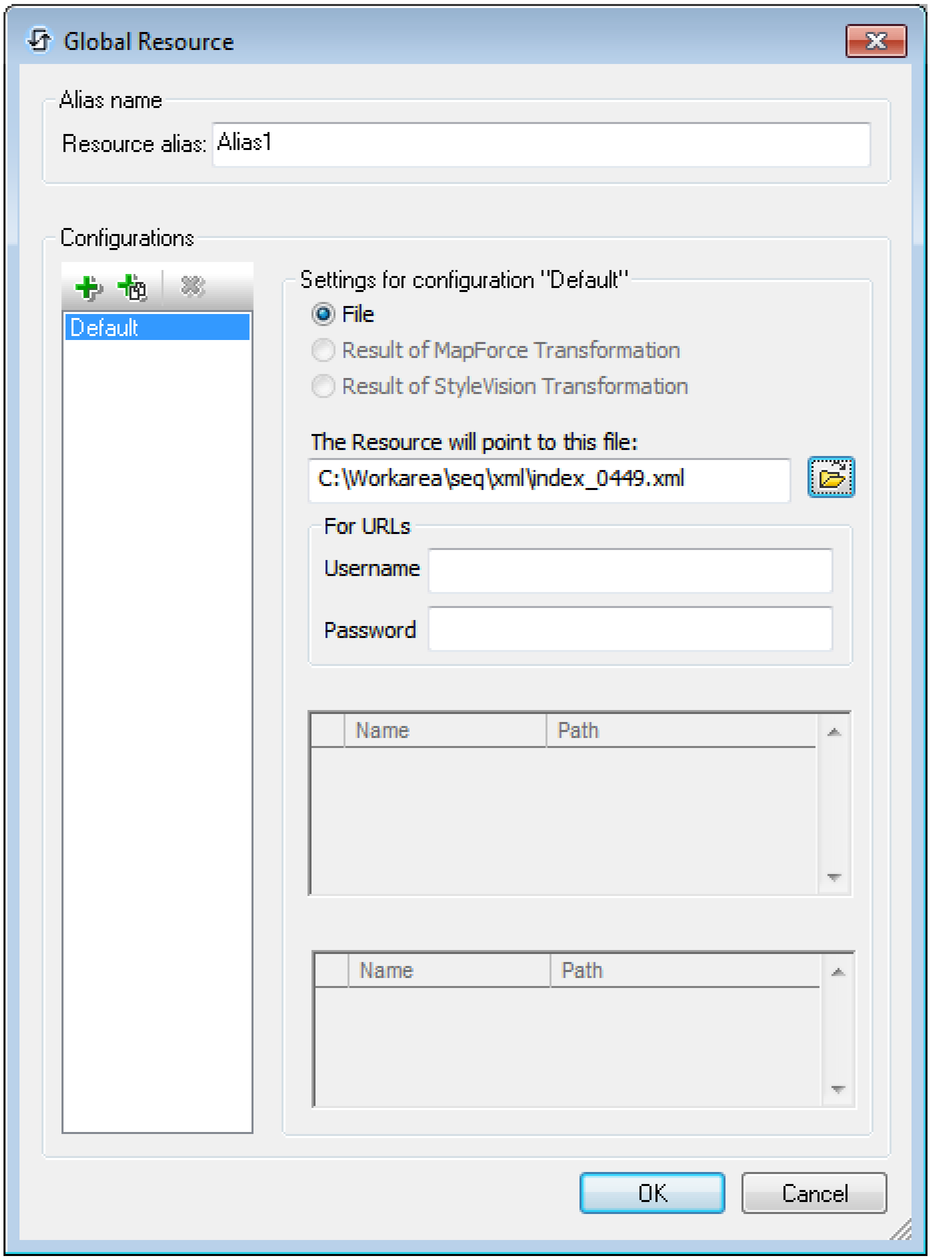
## 16.1.1 Files

The Global Resource dialog for Files (*screenshot below*) is accessed via the **Add | File** command in the Manage Global Resources dialog[1335]. In this dialog, you can define configurations of the alias that is named in the *Resource Alias* text box. After specifying the properties of the configurations as explained below, save the alias definition by clicking **OK**.

After saving an alias definition, you can add another alias by repeating the steps given above (starting with the **Add | File** command in the Manage Global Resources dialog[1335]).

### Global Resource dialog

An alias is defined in the Global Resource dialog (*screenshot below*).



Resource alias: Alias1

Default

Settings for configuration "Default"

● File

○ Result of MapForce Transformation

○ Result of StyleVision Transformation

The Resource will point to this file: C:\Workarea\seq\xml\index_0449.xml

For URLs

Username

Password

## Global Resource dialog icons

*Add Configuration:* Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.

*Add Configuration as Copy:* Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.

*Delete:* Deletes the selected configuration.

📂 *Open:* Browse for the file to be created as the global resource.

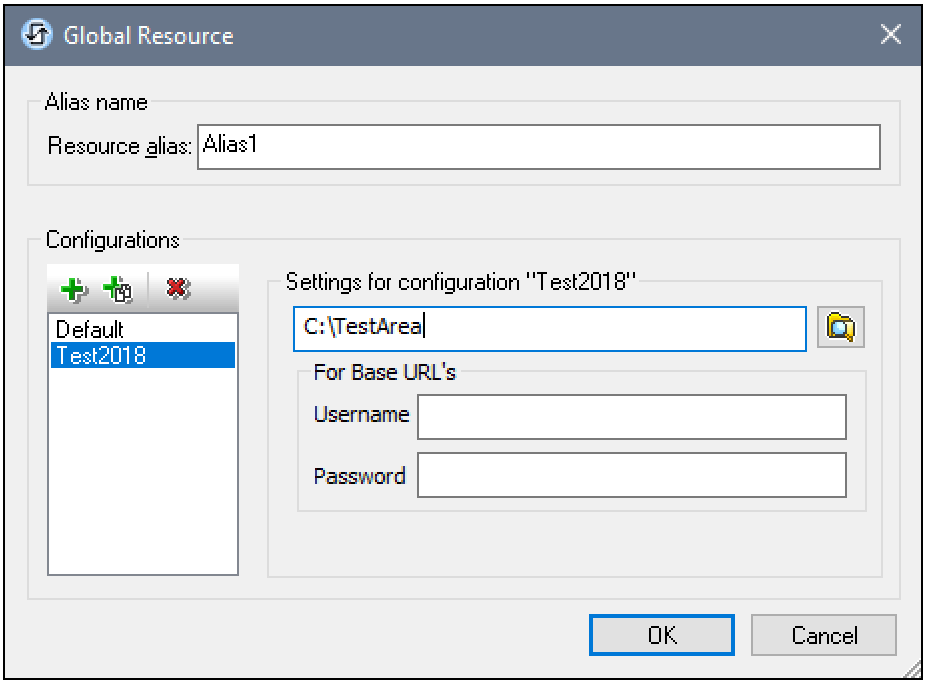
## Defining the alias

Define the alias (its name and configurations) as follows:

1. *Give the alias a name:* Enter the alias name in the *Resource Alias* text box.
2. *Add configurations:* The Configurations pane will have, by default, a configuration named `Default` (*see screenshot above*), which cannot be deleted or renamed. You can add as many additional configurations as you like by: (i) clicking the **Add Configuration** or **Add Configuration as Copy** icons, and (ii) giving the configuration a name in the dialog that pops up. Each added configuration will be shown in the Configurations list. In the screenshot above, two additional configurations, named `Long` and `Short`, have been added to the Configurations list. The Add Configuration as Copy command enables you to copy the selected configuration and then modify it.
3. *Select a resource type for each configuration:* Only a File resource may be selected as a global resource for a MobileTogether design.
4. *Select a file for the resource type:* If the resource is a directly selected file, browse for the file in the *Resource File Selection* text box.
5. *Define multiple configurations if required:* You can add more configurations and specify a resource for each. Do this by repeating Steps 3 and 4 above for each configuration. You can add a new configuration to the alias definition at any time.
6. *Save the alias definition:* Click **OK** to save the alias and all its configurations as a global resource. The global resource will be listed under Files in  the Manage Global Resources dialog[1335].

## 16.1.2 Folders

In the Global Resource dialog for Folders (*screenshot below*), add a folder resource as described below.



Global Resource

Alias name

Resource alias: Alias1

Configurations

Default
Test2018

Settings for configuration "Test2018"

C:\TestArea

For Base URL's

Username

Password

### Global Resource dialog icons

*Add Configuration:* Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.

*Add Configuration as Copy:* Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.

*Delete:* Deletes the selected configuration.

*Open:* Browse for the folder to be created as the global resource.

### Defining the alias

Define the alias (its name and configurations) as follows:

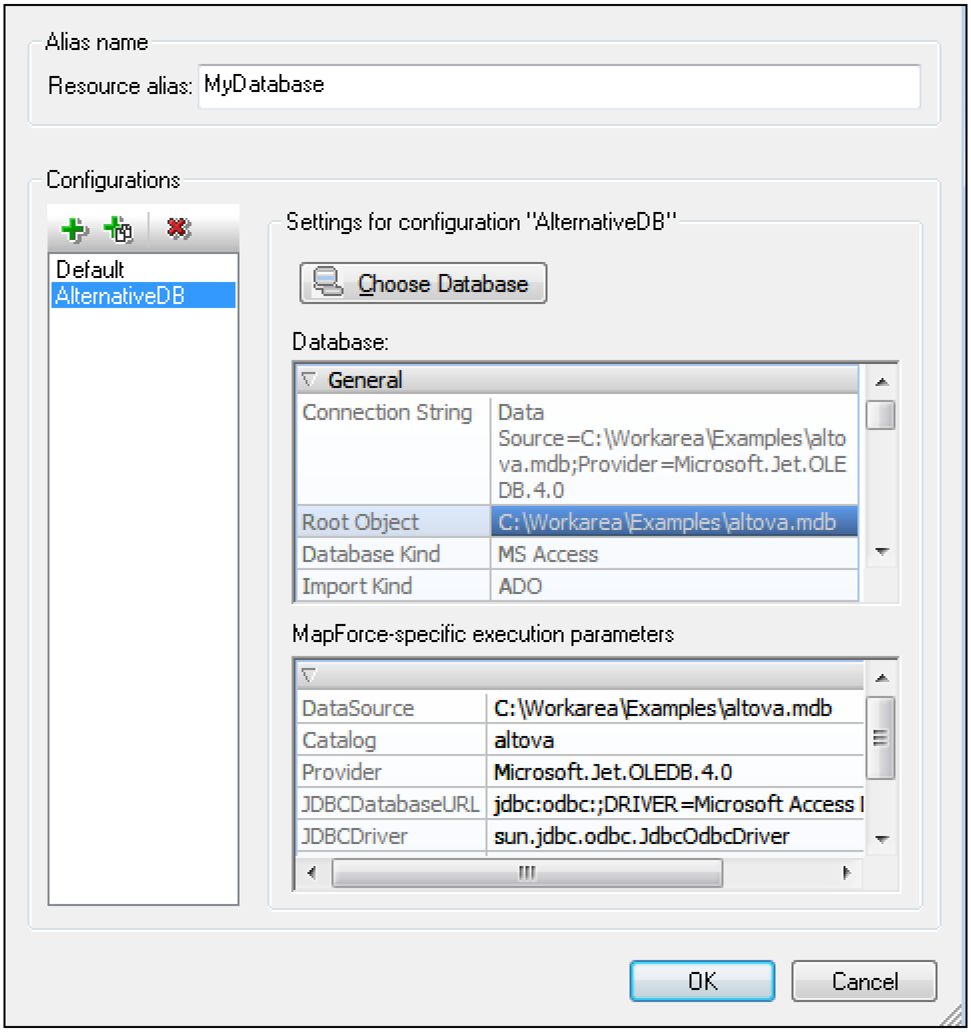
1. *Give the alias a name:* Enter the alias name in the *Resource Alias* text box.
2. *Add configurations:* The Configurations pane will have a configuration named Default (*see screenshot above*). This Default configuration cannot be deleted nor have its name changed. You can enter as many additional configurations for the selected alias as you like. Add a configuration by clicking the **Add Configuration** or **Add Configuration as Copy** icons. In the dialog which pops up, enter the

configuration name. Click **OK**. The new configuration will be listed in the Configurations pane. Repeat for as many configurations as you want.

3. *Select a folder as the resource of a configuration:* Select one of the configurations in the Configurations pane and browse for the folder you wish to create as a global resource. If security credentials are required to access a folder, then specify these in the *Username* and *Password* fields.
4. *Define multiple configurations if required:* Specify a folder resource for each configuration you have created (that is, repeat Step 3 above for the various configurations you have created). You can add a new configuration to the alias definition at any time.
5. *Save the alias definition:* Click **OK** in the Global Resource dialog to save the alias and all its configurations as a global resource. The global resource will be listed under Folders in the Manage Global Resources dialog[1335].

## 16.1.3 Databases

In the Global Resource dialog for Databases (*screenshot below*), you can add a database resource as follows:



Alias name

Resource alias: MyDatabase

Configurations

Default
AlternativeDB

Settings for configuration "AlternativeDB"

Database:

| General | |
|---|---|
| Connection String | Data Source=C:\Workarea\Examples\altova.mdb;Provider=Microsoft.Jet.OLEDB.4.0 |
| Root Object | C:\Workarea\Examples\altova.mdb |
| Database Kind | MS Access |
| Import Kind | ADO |

MapForce-specific execution parameters

| | |
|---|---|
| DataSource | C:\Workarea\Examples\altova.mdb |
| Catalog | altova |
| Provider | Microsoft.Jet.OLEDB.4.0 |
| JDBCDatabaseURL | jdbc:odbc:;DRIVER=Microsoft Access I |
| JDBCDriver | sun.jdbc.odbc.JdbcOdbcDriver |

## Global Resource dialog icons

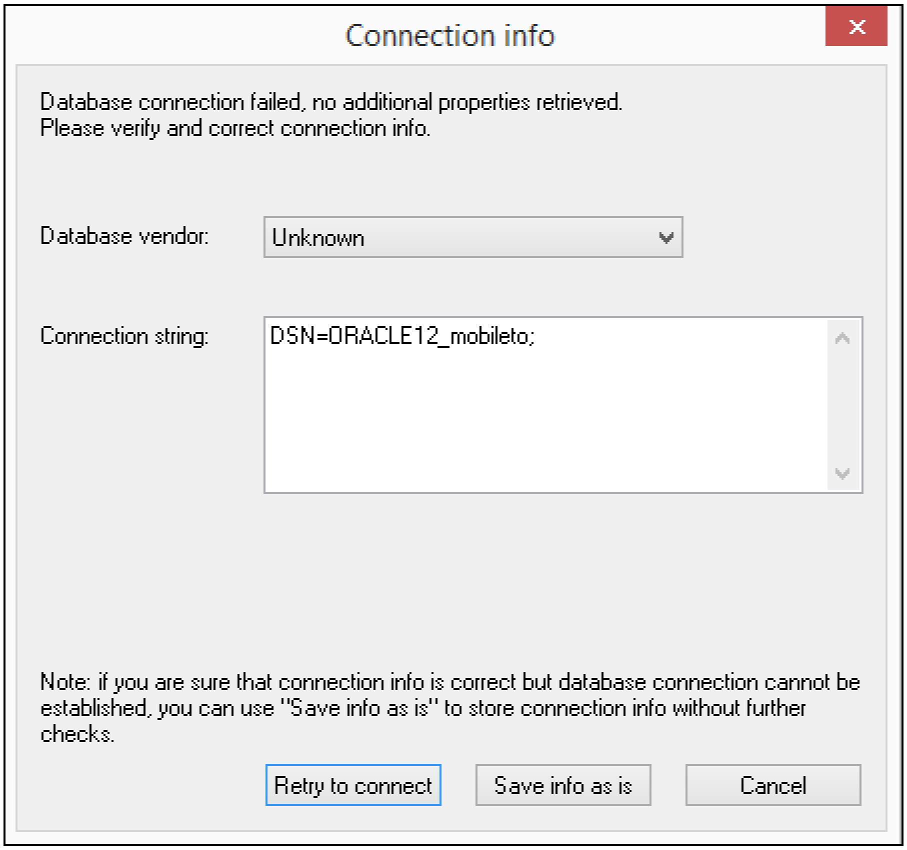
*Add Configuration:* Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.

*Add Configuration as Copy:* Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.

*Delete:* Deletes the selected configuration.

## Saving unverified connection info

If the connection information you enter cannot correctly access the database you want, then this might be because the connection information is correct only when the solution is deployed to the server but does not work from the local machine. If the connection does not work from the local machine (on which you are defining the global resource), then the Connection Info dialog (*screenshot below*) appears.



Connection info

Database connection failed, no additional properties retrieved.
Please verify and correct connection info.

Database vendor: Unknown

Connection string: DSN=ORACLE12_mobileto;

Note: if you are sure that connection info is correct but database connection cannot be established, you can use "Save info as is" to store connection info without further checks.

You now have the following options:

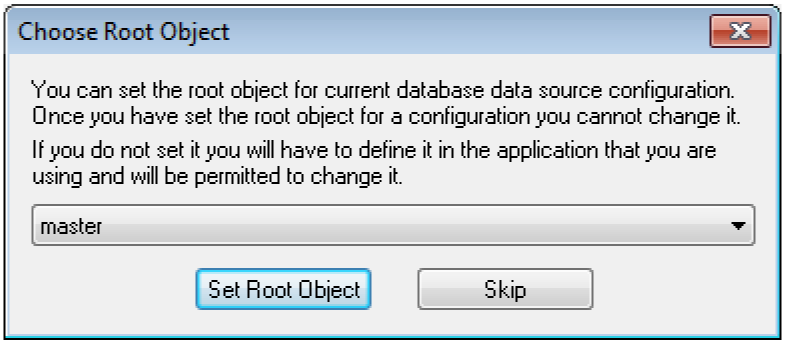
- *Retry to connect:* Enter the connection information that will enable you to connect from the local machine, and click **Retry to Connect**. MobileTogether will attempt to make the connection.

- *Save info as is:* Saves the connection information without attempting to connect or verify the connection info. This connection info will be used when the solution is deployed to the server.
- *Cancel:* Cancels the process of defining a database as a global resource.

## Defining the alias

Define the alias (its name and configurations) as follows:

1. *Give the alias a name:* Enter the alias name in the *Resource Alias* text box.
2. *Add configurations:* The Configurations pane will have a configuration named Default (*see screenshot above*). This Default configuration cannot be deleted nor have its name changed. You can enter as many additional configurations for the selected alias as you like. Add a configuration by clicking the **Add Configuration** or **Add Configuration as Copy** icons. In the dialog which pops up, enter the configuration name. Click **OK**. The new configuration will be listed in the Configurations pane. Repeat for as many configurations as you want.
3. *Start selection of a database as the resource of a configuration:* Select one of the configurations in the Configurations pane and click the **Choose Database** icon. This pops up the Create Global Resources Connection dialog.
4. *Connect to the database:* Select whether you wish to create a connection to the database using the Connection Wizard, an existing connection, an ADO Connection, an ODBC Connection, or JDBC Connection.
5. *Select the root object:* If you connect to a database server where a root object can be selected, you will be prompted, in the Choose Root Object dialog (*screenshot below*), to select a root object on the server. Select the root object and click **Set Root Object**. The root object you select will be the root object that is loaded when this configuration is used.

   You can set the root object for current database data source configuration. Once you have set the root object for a configuration you cannot change it.

   If you do not set it you will have to define it in the application that you are using and will be permitted to change it.

   master

   If you choose not to select a root object (by clicking the **Skip** button), then you can select the root object at the time the global resource is loaded.

6. *Define multiple configurations if required:* Specify a database resource for any other configuration you have created (that is, repeat Steps 3 to 5 above for the various configurations you have created). You can add a new configuration to the alias definition at any time.
7. *Save the alias definition:* Click **OK** in the Global Resource dialog to save the alias and all its configurations as a global resource. The global resource will be listed under databases in the Manage Global Resources dialog.

## 16.2 Using Global Resources

There are several types of global resources (file-type, folder-type , and database-type). Some scenarios in which you can use global resources in MobileTogether Designer are listed here: Files and Folders 1344 and Databases 1345.

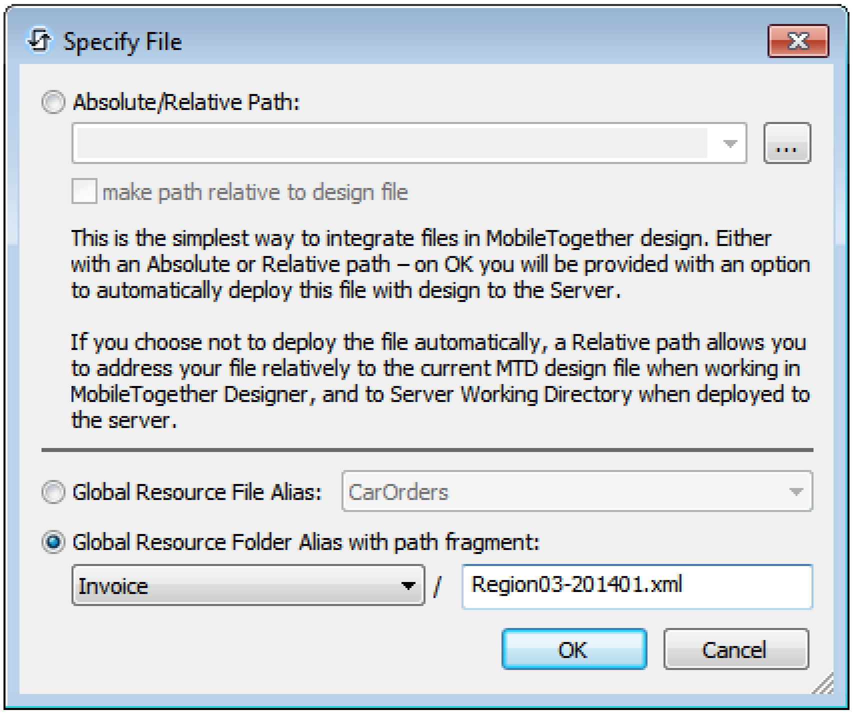
### Selections that determine which resource is used

There are two application-wide selections that determine what global resources can be used and which global resources are actually used at any given time:

- *The active Global Resources XML File* is selected in the Global Resource dialog 1335. The global-resource definitions that are present in the active Global Resources XML File are available to all files that are open in the application. Only the definitions in the active Global Resources XML File are available. The active Global Resources XML File can be changed at any time, and the global-resource definitions in the new active file will immediately replace those of the previously active file. The active Global Resources XML File therefore determines: (i) what global resources can be assigned, and (ii) what global resources are available for look-up (for example, if a global resource in one Global Resource XML File is assigned but there is no global resource of that name in the currently active Global Resources XML File, then the assigned global resource (alias) cannot be looked up).
- *The active configuration* is selected via the menu item **Tools | Active Configuration** 1654 or via the Global Resources toolbar. Clicking this command (or drop-down list in the toolbar) pops up a list of configurations across all aliases. Selecting a configuration makes that configuration active application-wide. This means that wherever a global resource (or alias) is used, the resource corresponding to the active configuration of each used alias will be loaded. The active configuration is applied to all used aliases. If an alias does not have a configuration with the name of the active configuration, then the default configuration of that alias will be used. The active configuration is not relevant when assigning resources; it is significant only when the resources are actually used.

## 16.2.1 Assigning Files and Folders

In certain usage scenarios, file-type and folder-type global resources can be used to specify the file or folder to use. For example, in the Page Sources Pane 270, the default file of a data source can be assigned via a global resource. In such scenarios, the Specify File (*screenshot below*) appears.



○ Absolute/Relative Path:

□ make path relative to design file

This is the simplest way to integrate files in MobileTogether design. Either with an Absolute or Relative path – on OK you will be provided with an option to automatically deploy this file with design to the Server.

If you choose not to deploy the file automatically, a Relative path allows you to address your file relatively to the current MTD design file when working in MobileTogether Designer, and to Server Working Directory when deployed to the server.

○ Global Resource File Alias: CarOrders

● Global Resource Folder Alias with path fragment:

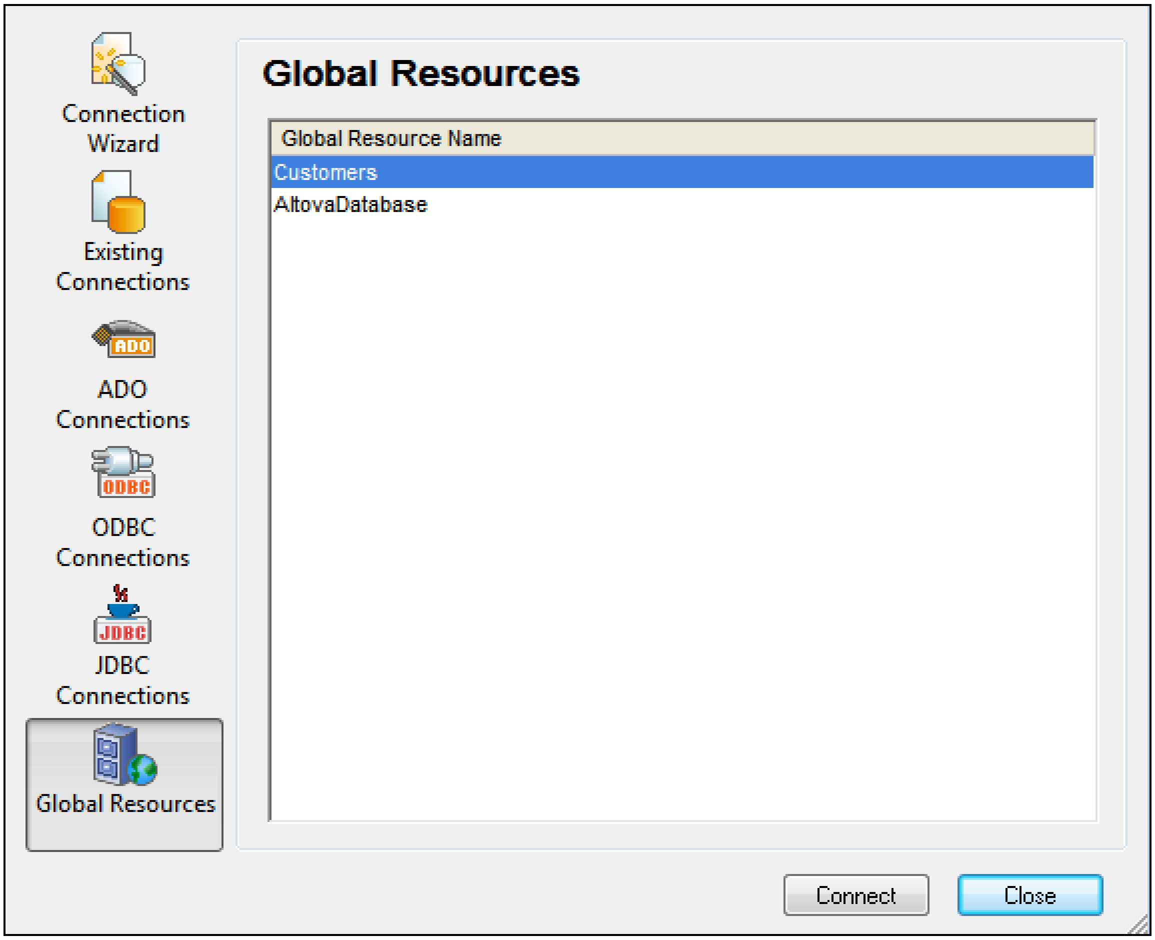
Invoice / Region03-201401.xml

Select whether you wish to use a file-type or folder-type global resource. The combo boxes display, respectively, all the file-type global resources and all the folder-type global resources that have been defined in the currently active Global Resources XML File[1335]. Select the required global resource. In the case of file-type global resources, the selected alias maps to a file. In the case of folder-type global resources, the selected alias maps to a folder, so you will have to enter the rest of the path to locate the resource (*see screenshot above*). Click **OK**.

If the selected global resource has more than one configuration, then the database resource for the currently active configuration is used (check **Tools | Active Configuration** or the Global Resources toolbar).

## 16.2.2 Assigning Databases

When a command is executed that imports data or a data structure from a DB, you can select the option to use a global resource (*screenshot below*).



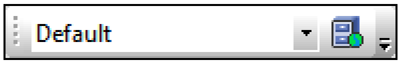
In the Connection dialog (*screenshot above*), all the database-type global resources that have been defined in the currently active Global Resources XML File[1335] are displayed. Select the required global resource and click **Connect**. If the selected global resource has more than one configuration, then the database resource for the currently active configuration is used (check **Tools | Active Configuration** or the Global Resources toolbar), and the connection is made.

## 16.2.3 Changing the Active Configuration

One configuration of a global resource can be active at any time. This configuration is called the active configuration, and it is active application-wide. This means that the active configuration is active for all global resources aliases in all currently open files and data source connections. If an alias does not have a configuration with the name of the active configuration, then the default configuration of that alias will be used. As an example of how to change configurations, consider the case in which a file has been assigned via a global resource with multiple configurations. Each configuration maps to a different file. So, which file is selected depends on which configuration is selected as the application's active configuration.

Switching the active configuration can be done in the following ways:

- Via the menu command **Tools | Active Configuration**. Select the configuration from the command's submenu.
- In the combo box of the Global Resources toolbar (*screenshot below*), select the required configuration.



In this way, by changing the active configuration, you can change source files that are assigned via a global resource.

# 17 Subprojects and Modules

This section describes two MobileTogether Designer features that are related:

- Subprojects 1349 are projects that can be included in other projects. Subprojects enable you to compose projects from multiple projects. You can therefore, for example, store commonly used design components in a subproject and reuse this subproject in multiple projects.
- Modules 1352 provide a way to group a set of design components. The components of a module can then be treated as a group. For example, to identify them visually as a group by giving them a single background color.

Subprojects and modules are related in a feature that enables you to extract a subproject from a project. Modules provide a way to specify what components of the project to export to the subproject.

# 17.1 Subprojects

You can include MobileTogether projects (**.mtd** files) as subprojects of your current project. The current project can then use components of the included subprojects. Conversely, the components of a subproject can be reused across multiple (parent) projects, saving you the need to redefine it in multiple projects. Note that you can also include a subproject that itself contains subprojects.

The following components can be defined in a subproject and reused in parent projects.

- Design pages 380 except top pages
- Control templates 1200
- Page sources 315 except the `$PERSISTENT` page source.
- DB sources 338
- Style sheets 1318 except the project style sheet
- Rich Text style sheets 1225
- Action groups 940
- User-defined XPath functions 1293
- User-defined global variables 1309
- Chart settings 1149
- Server action libraries 1551
- Localization strings and languages 308
- Modules 1352
- ID Push Notification 1125 button sets
- OAuth settings 332
- Namespaces 310
- InApp Purchase 1502 products

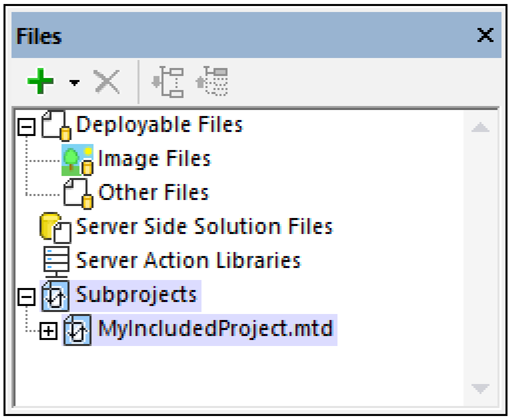
## Working with subprojects

There are two steps involved in using subprojects:

1. Create a subproject 1350 (new or from an existing project).
2. Include a subproject 1351 in your current project.

These steps are described in the subsections of this section.

All the included subprojects of a project are displayed in the project's Files Pane 259 (*see screenshot below*).



The Files Pane[259] acts as the control panel for displaying, including, importing, and removing subprojects.

*Deploying to server and generating AppStore Apps*
If your project contains subprojects and you deploy it to the server[291] or export it as an AppStore App[1471], all subprojects are merged to the main project.

### Subprojects: including v/s copying

When a subproject is **included**, it is referenced from the main project. If you modify the subproject, then all the changes you make in the subproject will automatically be reflected in all the projects that include the subproject. The included subprojects of a project are listed in the project's Files Pane[259], under the *Subprojects* item (*see screenshot above*).

On the other hand, when a subproject is **copied** into a main project, its components are incorporated directly in the main project as components of the main project. The copy action is carried out via the Files Pane[259] (*see screenshot above*), by right-clicking the included subproject that you want to copy and selecting the command **Include Subproject as a Copy**. The included subproject's components will be copied into the main project as the latter's components and the reference to the subproject will be removed from the Files Pane[259]. The subproject (`.mtd` file) itself will not be deleted and can be included in any project at a later time.

## 17.1.1 Create a Subproject

There are two ways in which you can create a subproject :

- The simpler way is to extract a subproject from an existing project with the **Refactor | Extract New Subproject**[1614] menu command. In this case all you need to do is to specify (i) the components you wish to extract from the current project to the subproject, and (ii) the location and name of the extracted subproject file. The selected components will be extracted into a new .mtd file and saved to the location you specified. This extracted subproject will automatically be referenced as a subproject of the current project and be displayed as a subproject in the project's Files Pane[259]. The extracted components will no longer exist in the current project, but will be referenced from the newly created subproject.

- Alternatively, you can create a new MobileTogether Designer project. To set it up as a subproject, do the following: (i) Add the components that you want to use in main projects; (ii) In the Modules Pane 263, configure the top pages to be not exported (since subprojects must not have any visible top pages). After you have created, configured, and saved your subproject, you will need to explicitly include it in the main projects 1351 in which you want to use it.

### How to set which components of a project to export

When you extract a subproject from a project, by default, all the components will be exported to the subproject. However, you might not want to export (or not export) a subset of all the components. In this case, do the following:

1. Assign the components that you want to export (or not export) to a module as described in the sections *Creating modules: module names and item names* 1352 and *Reassign modules to other modules* 263.
2. Set the *Export* property of the module to *Exported* or *Not Exported* as appropriate.
3. Extract the subproject with the **Refactor | Extract New Subproject** 1614 menu command.

**Note:** The *Export* property can be set also on the *Unassigned Items* module.

**Note:** Since top pages must not be present in a subproject, you can set the module/s containing the top page/s to not be exported.

## 17.1.2 Include a Subproject

You can include a MobileTogether project as a subproject in another MobileTogether project. This is useful if you want to reuse components from the subproject in the main project in which you are including it. (For a list of components that can be defined in subprojects, see the topic Subprojects 1349.) A subproject can therefore contain components that you use commonly across multiple projects.

To include a subproject in your current project, do one of the following:

- Select the menu command **Refactor | Include Subproject** 1616. In the Open dialog that appears, browse for the project you want to add as a subproject, select it, and click **Open**.
- In the Files Pane 259 of the project in which you want to include a subproject, right-click the **Subprojects** item and, in the context menu that appears select the command **Include Subproject**. In the Open dialog that appears, browse for the project you want to add as a subproject, select it, and click **Open**.

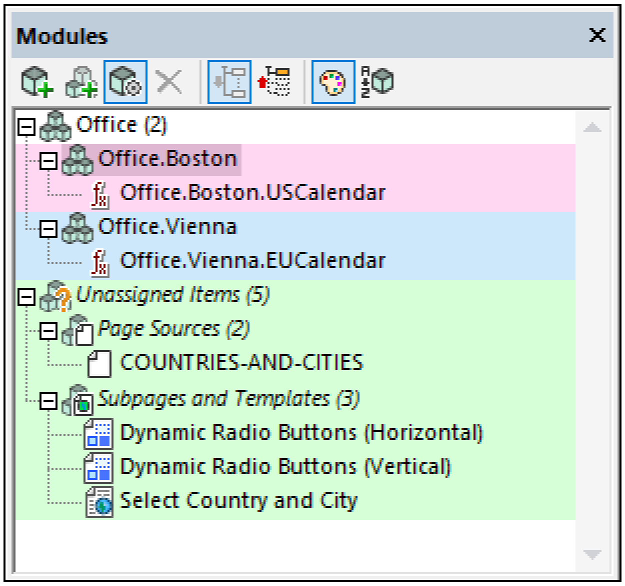
The subproject will be added to the current project. It will be displayed in the Files Pane 259 and can now be managed there. For example, you can copy the components of an included project to the current project. See the description of the Files Pane 259 for more information.

# 17.2 Modules

The Modules feature of MobileTogether Designer enables you to group components of a MobileTogether project into different modules and assign properties to individual modules. This enables you to group project components and to then conveniently apply a common property to the whole group of components. When project components have been grouped together in a module, we refer to them as the items of that module. All the modules of a project and their items can be viewed and managed in the Modules pane 263 (*screenshot below*).

The following properties can be set on a module. The selected property value will apply to all items of the module.

- *Background color*: This enables you to easily locate all the components of a module in application windows and dialogs (because all components of a module will have the same background color).
- *Export setup*: You can specify whether the components of a module should be exported or not when a subproject is created from a project 1350. You can therefore group those components that you want to export into a module and set the module to the *Not Exported* property.



## Module items

The following project components can be assigned as the items of a module:

- Action Groups 940
- Pages 381
- Subpages 381
- Page Sources (Data Sources) 315
- Control Templates 1200
- User-Defined XPath/XQuery Functions 1293
- User Variables 1309

Within each module (other than the *Unassigned Items* module), the module items are listed alphabetically in a flat list. In the *Unassigned Items* module, module items are grouped first by component type and are sorted alphabetically within each type.

## Creating modules: module names and item names

A module is created in the Modules pane[263] via the **Add Module** and **Add Submodule** commands. (A submodule is defined as a module that is within another module.) A module must exist—that is, be created explicitly—before any item can be added to it.

*Module names*

When you create a module, you must give it a name. The name of a module or submodule is a string that does not contain a period. For example, in the screenshot of the Modules pane[263] shown above, note the names of the *Office* module and its two submodules, *Boston* and *Vienna*.

*Item names*

A module **item**, however, has a name that is composed of the module name followed by a period and the component name (for example, **`ModuleName.ComponentName`**). If the item belongs to a submodule, then the **`ModuleName`** part of the item's name contains the names of ancestor modules, with each ancestor being separated from the next by a period. For example, in the screenshot above, see the name of the function in the *Boston* module (**`Office.Boston.USCalendar`**). The item's name consists of the names of its two ancestor modules separated by a period.

The importance of an item (or component) name is that it determines whether it belongs to a module and to which one. For example, if a component (such as a function) is created in the project and is given a name that contains a module name as described in the previous paragraph, then that component will be added automatically to the list of items of that module in the Modules pane[263] (*see screenshot above*). If a component name does not contain a match for any module or submodule, then it is listed as an item of the *Unassigned Items* module—which exists by default and does not need to be explicitly created.

If a component name contains periods and if the module hierarchy indicated by those periods does not exist, then that module hierarchy is created for the component and the component is correctly placed in the hierarchy. If a module is created automatically by the system in this way—as opposed to having been created explicitly by you—then it is removed automatically if it becomes empty and has no property.

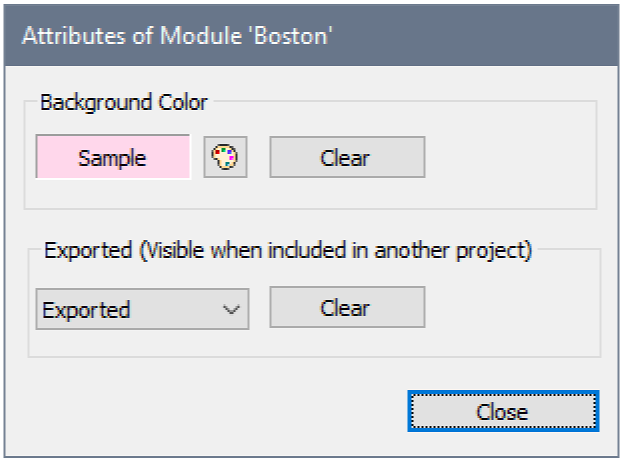
If you wish to move an item from one module to another, then you can do one of two things:

- Drag it from its current module and drop it in the new module. As a result, the module name part of the item's name will be changed automatically.
- Rename the item (via its context menu) to have a module name part that matches an existing module. As a result, the item will be moved automatically to the new module.

**Note:** A component can belong to one module only (either the *Unassigned Items* module or one that you create).

## The Background Color property

When you right-click a module (including the *Unassigned Items* module), you can select, in the context menu that appears, the **Module Settings** command to display the Attributes (or Properties) dialog of the current module (*screenshot below*, which shows the Attributes dialog of the *Boston* module).


Attributes of Module 'Boston'
Background Color
Sample
Clear
Exported (Visible when included in another project)
Exported
Clear
Close

By setting the background color of the module, you set the background color of all the items of this module. This enables you to easily locate the corresponding project components in application windows and dialogs—because all components of the module will have the same background color. Note that the background color of a module also applies to all its descendant submodules unless a descendant module has its own background color setting.

You can toggle background colors on/off via a toolbar button of the Modules pane[263].

## The Export property

The *Export* property is available in the Attributes dialog of a module (*see screenshot above*), which is accessed via the **Module Settings** command in the context menu of the module[263].

The property specifies whether the components of that module will be exported or not when the project is extracted to a subproject[1350] (via the **Refactor | Extract New Subproject**[1614] command). The ability to export or not export components is useful when extracting subprojects because you might want to export only a few specific components, or you might want to not export one or more components (for example, a top page, since top pages are not allowed in subprojects). Note that the *Export* property can also be used for the *Unassigned Items* module.

Select the value of the *Export* property: *Not Set* (which causes the value of the parent module to be inherited), *Exported*, or *Not Exported*.

## Application windows and dialogs that display modules

The following windows and dialogs provide display module-related features:

- Page Sources Pane[270]
- Pages Pane[257] (showing pages, sub pages, and control templates)
- XPath/XQuery Window[1244]
- Static Global Variables[1300] (lower part for user variables)
- Action Groups[940]

Right-click a module in these windows and dialogs to access module-related commands.

# 18 Simulation

You can simulate the workflow on the currently selected preview device[254]. To preview (simulate) a solution on a different device, change the preview device[254].

The following simulation methods are available for running and testing a solution:

- Simulation in MobileTogether Designer[1356]: A quick way to test whether the design is free from errors.
- Simulation on Server[1362]: Additional to testing that the design is error-free, this simulation enables you to test whether data sources are correctly located, whether URLs are correct, whether current server settings are appropriate, and whether the server has all permissions to access the used DBs, URLs, and files.
- Simulation on Client[1370]: Enables you to test the actual client-server interactions in a trial run. In this simulation, MobileTogether Designer plays the role of the server.

**Note:** You can also deploy a solution to the server, and, in the *Workflows* tab of the Web UI of MobileTogether Server, you can click a solution to run it in the web browser.

While the simulation progresses, the Messages Pane[1385] of the MobileTogether Designer GUI provides a detailed and step-by-step report of all the activities taking place. This is an extremely important feature for testing and debugging design files.

**Note:**

- If you run a designer or server simulation of a design containing geolocation actions, then you will need to define the geolocations to use in the simulation. How to do this is described in the section Geolocation Settings[1372].
- Since the simulators, which run on desktops, are not NFC-enabled, an NFC sample file can be used to simulate an NFC tag. See the section NFC Sample Files[1377] for a template of an NFC sample file and how to use sample files to simulate the reading of NFC data.
- When simulating push notifications in which the sending and receiving solutions are different, data in a simulated sent PN can be recorded to a file. This data can be loaded into a simulation of the receiving solution so that a PN can be simulated as having been received.
- To simulate a device's address book, a sample XML file of the contacts of an address book can be used. See Contacts Sample Files[1381] for how to create and use such a file.

## Simulation device

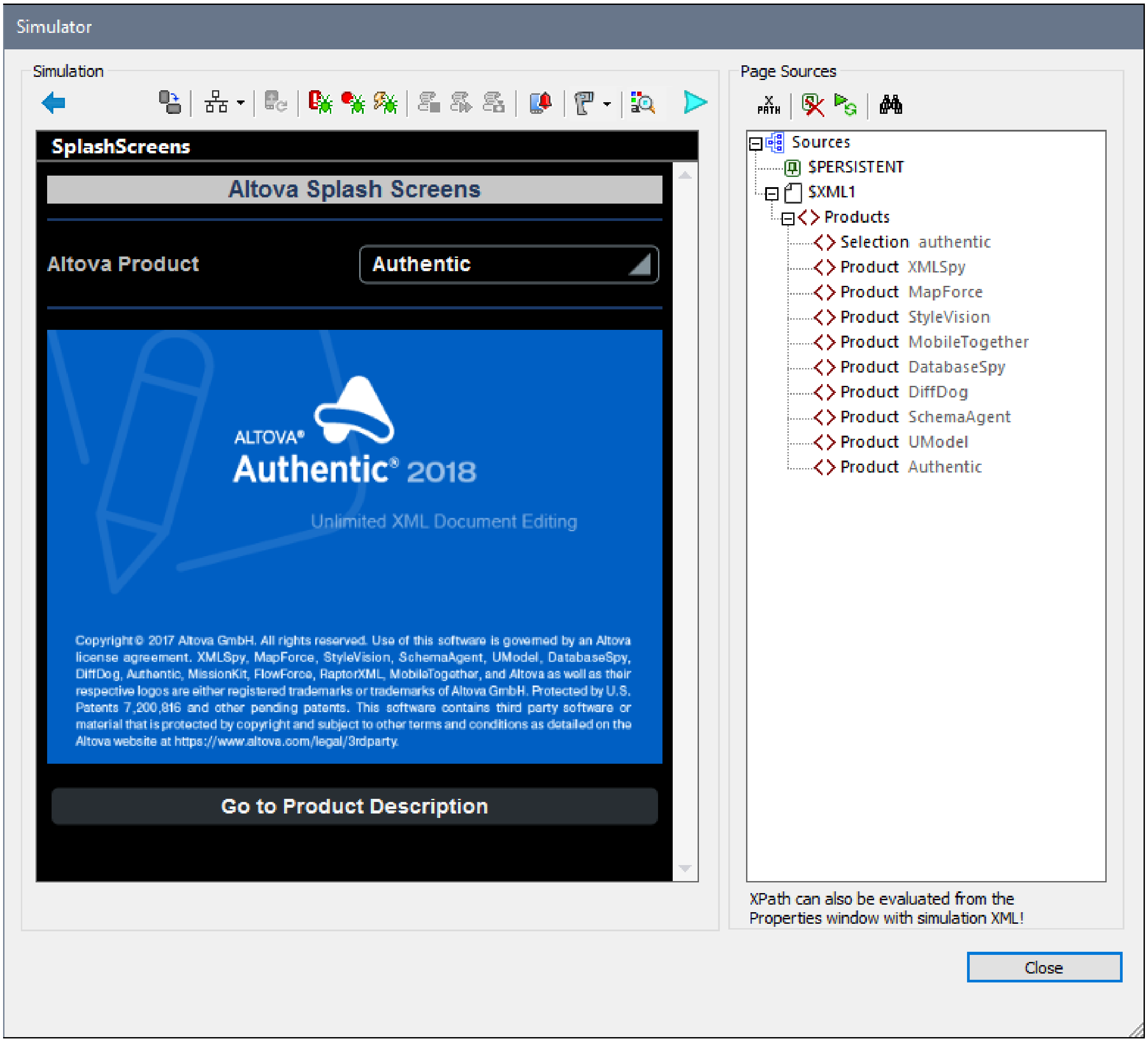
You can select the device that you want to simulate in the *Preview Device*[254] combo box. To see a simulation on a different platform or different device, choose the new device and re-start the simulation.

## Simulation language

The simulation language for designer[1356] and server[1362] simulations is selected via the **Project | Simulation Language**[1606] command. The language of client simulations[1370] is the same as the language of the client mobile device on which the simulation runs.

# 18.1 Simulation in MobileTogether Designer

You can run a simulation of the project workflow directly within MobileTogether Designer. The simulation device will be the device currently selected in the Preview Device combo box of the Main toolbar. You can change the preview device to see the simulation on different devices. To run the simulation, select **Run | Simulate Workflow** or **F5**. This opens the simulator and starts the simulation. Simulation in the Designer reports both server and client messages in the Messages Pane[1385].



Altova Product: Authentic

Copyright© 2017 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,200,816 and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at https://www.altova.com/legal/3rdparty.

```
Sources
  $PERSISTENT
  $XML1
    Products
      Selection  authentic
      Product  XMLSpy
      Product  MapForce
      Product  StyleVision
      Product  MobileTogether
      Product  DatabaseSpy
      Product  DiffDog
      Product  SchemaAgent
      Product  UModel
      Product  Authentic
```

XPath can also be evaluated from the Properties window with simulation XML!

## Simulation language

The simulation language for designer[1356] and server[1362] simulations is selected via the **Project | Simulation Language**[1606] command. The language of client simulations[1370] is the same as the language of the client mobile device on which the simulation runs.

## File locations

When a simulation is run directly in MobileTogether Designer, file locations are resolved exactly as specified in the design. Relative paths are relative to the design file's location. Compare these locations to how file locations are resolved when using the server for workflow simulation[1362].

## Simulator features

The simulator window provides the following features:

- The left-hand (Simulation) pane displays the simulation. Options for the Simulation pane are described below.
- The right-hand (Page Sources) pane shows the changes taking place in the XML data as the simulation progresses. Options for the Page Sources pane are described below.
- While the simulation progresses, the Messages Pane[1385] of the MobileTogether Designer GUI provides a detailed and step-by-step report of all the activities taking place. You can therefore see what is happening at each step in the progress of the workflow. This is an invaluable feature for testing and debugging design files.
  - *Simulations on the designer report activities on both server and client.*
  - *Simulations on the server report client messages.*
  - *Simulations on the client report server messages*.
- Controls that require user interaction are enabled. In the screenshot above, for example, the combo box is enabled.

*Simulation pane toolbar*

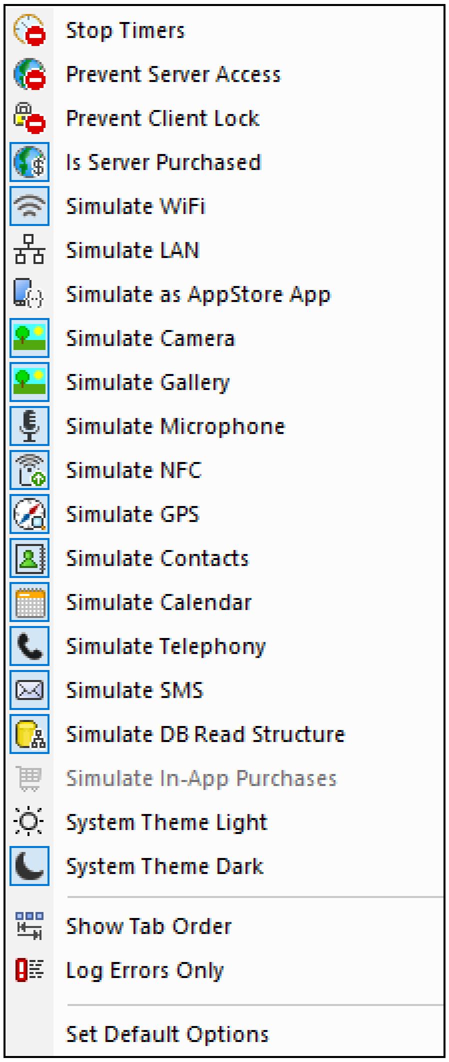
The toolbar of the Simulation pane contains the following buttons (from left to right):

- *Back:* If the page is a subpage[257], then clicking **Back** closes the subpage. If the page is a top page[257], the simulator closes. Also see **`OnBackButtonClicked`**[393].
- *Change Orientation:* You can switch the view between landscape and portrait.
- *Simulator Options:* Drops down a menu of simulator options (described below under *Simulator options*).
- *Simulate Return to App (Reopen):* Enabled when page refreshes have been defined to occur on page reopens[390]. Refreshes the page during simulations.
- *Stop at Next Error:* Stops the simulation at the next XPath error and displays the XPath expression in the XPath Debugger[1388].
- *Stop at Next Breakpoint:* Stops the simulation at the next breakpoint (which may be on an action or XPath expression) and opens the appropriate Debugger[1388].
- *Stop at Next Action:* Stops the simulation at the action/s of the next event that is triggered and displays the action/s in the Actions Debugger[1390].
- *Stop Recording Test Case:* When the recording of a test case[1401] is started, the simulator is opened and the recording runs in the simulator. Clicking this button: (i) stops the recording of the test case[1401], and (ii) opens the Recorded Test Case Confirmation dialog, in which you specify the name of the recorded test case.
- *Playback Next Step:* Plays back the next step in the currently running test-case playback[1403]. This button is enabled when playback is set to step-by-step. See the *Playback Options* section of Playing Back a Test Case[1403].
- *Record Snapshot:* Records a snapshot of the test case that is being recorded[1401]. This button is enabled when the recording of snapshots has been set to manual (not automatic). See the *Recording Options* section of Recording a Test Case[1401].
- *Push Notifications:* If the design contains a push notification, opens the Manage Recorded Push Notification (Simulation) dialog. See Simulating Push Notifications[1133] for a description of usage.

- *Barcode Scanner:* Drops down a menu that has three parts, one for each supported barcode scanner[1145]. For each scanner, you can simulate receiving barcode data from the scanner. In the case of Zebra scanners, you can also simulate connection established, receiving image data, and connection terminated. Note that (for Zebra scanners) you must simulate a connection-establishment before the options for subsequent simulations become available. For some of these simulations, a dialog will appear in which you must supply the data for the simulation (for example an ISBN or an image file).
- *Styles Inspector:* Open the Styles Inspector window[1332].
- *Purchase:* If the design enables in-app purchases, provides a list of all the in-app purchase products that are available for purchase and enables you to purchase one or more. The list of products is obtained from the In-app Purchase XML data file for simulations[1669]. See In-App Purchases[1502] for more information.
- *Purchased:* If the design enables in-app purchases, provides a list of all the in-app purchase products that are listed as purchased (that is, as a **`Purchase`** element) in the In-app Purchase XML data file for simulations[1669]. If you select one of these purchased products, then this product together with all its purchase properties is considered, in the simulation, to have been purchased. If you were to subsequently run the Query Purchases[934] action or Restore Purchases[934] action, then the products that you selected will be returned (by the action) as having been purchased. See In-App Purchases[1502] for more information.
- *Simulate Reception of MQTT Message:* If the design enables the receipt of MQTT messages, simulates actions that are performed on receiving an MQTT message. The MQTT messages to simulate are taken from a sample file specified in the Options dialog[1669].
- *Simulate Reception of Broadcast Message:* If the design enables the receipt of Broadcast messages, simulates actions that are performed on receiving a broadcast message. The broadcast messages to simulate are taken from a sample file specified in the Options dialog[1669].
- *Submit:* If the page is not the last page, then clicking **Submit** takes you to the next page. If the page is the last page, then the workflow is exited. Also see **`OnSubmitButtonClicked`**[394].
- *Refresh Page:* If the Page Refresh option[390] has been set to *Manual*, then this button becomes visible. On clicking **Refresh Page**, the page is updated with the changed data.

*Simulator options*

On clicking the **Simulator Options** toolbar button, a dropdown list of options for the simulator is displayed *(screenshot below)*. Each option is described below. Default settings are specified in the Simulation 1[1668] and Simulation 2[1669] tabs of the Options dialog. Note that, if you change a setting in the simulator, the new setting becomes the default and will be shown as such in the Simulation 1[1668] and Simulation 2[1669] tabs of the Options dialog.



- *Stop Timers:* If a timer has been set to run at intervals[390] and actions have been defined to be executed at these intervals, you can stop timers (and, consequently, actions) by clicking **Stop Timers**. This will clear up the clutter of messages generated by these actions, and will allow you to more easily analyze other messages and aspects of the workflow.
- *Prevent Server Access:* When selected, disables access to the server, and so allows you to test the solution's behavior in a server-connection-error scenario. When unselected, server access is allowed. For more information about this feature, see Server Connection Errors[394].
- *Prevent Client Lock:* When selected, prevents the client being locked from server access when the Lock Client[919] action is executed. If the server is inaccessible because the Lock Client[919] action has been executed from another client, then preventing the lock will of course not work.
- *Is Server Purchased:* For simulations in the designer and for trial runs on client, simulates that MobileTogether Server licenses have been purchased. For simulations on the server, the actual purchase-state of licenses on the server is returned.
- *Simulate WiFi:* Sets the `mt-connected-via-wifi`[1262] XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This allows the simulator to behave as though WiFi access is available. In this way, you can simulate design scenarios in which WiFi access is required.
- *Simulate LAN:* Sets the `mt-connected-via-lan`[1262] XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This allows the simulator to behave as though a LAN connection

is available. As a result, you can simulate design scenarios that require a LAN connection.

- *Simulate as AppStore App:* Sets the static global variable **`MT_IsAppStoreApp`**[1300] to `true()` when enabled, to `false()` when disabled. This allows simulations to be carried out that are conditional on the value of this variable.
- *Simulate Camera:* When toggled on, the simulator behaves as though the device's camera is available. This enables you to simulate design scenarios that require camera access.
- *Simulate Gallery:* When toggled on, the simulator behaves as though the device's photo gallery is available. This enables you to simulate design scenarios that require gallery access.
- *Simulate Microphone:* When toggled on, the simulator behaves as though the device's microphone is available. This enables you to simulate design scenarios that require microphone access.
- *Simulate NFC:* When selected, enables NFC functionality so that NFC actions can be executed. Actual NFC data is supplied to the simulator via NFC sample files[1377].
- *Simulate Bluetooth (BT):* When toggled on, the simulator behaves as though the device's Bluetooth connectivity is available. This enables you to simulate Barcode Scanner scenarios[1145] that require Bluetooth.
- *Simulate GPS:* When selected, enables geolocation functionality so that geolocation features can be tested. Dummy geolocations can be supplied via the Geolocations XML file[1372], which is used specifically to supply geolocations for simulations.
- *Simulate Contacts:* When toggled on, the simulator behaves as though the device's address book is available. This enables you to simulate design scenarios that require access to the address book. The address book is simulated either from a sample file[1381] or from your Microsoft Outlook contacts[1663]. Which option to use is specified in the Simulation 2 tab of the Options dialog[1669].
- *Simulate Calendar:* When toggled on, the simulator behaves as though the device's calendar is available. This enables you to simulate design scenarios that require access to the calendar. The calendar is simulated either from a sample file[1382] or from your Microsoft Outlook calendar[1663]. Which option to use is specified in the Simulation 2 tab of the Options dialog[1669].
- *Simulate Telephony:* When toggled on, the simulator behaves as though the device's telephony functionality is available. This enables you to simulate design scenarios that require telephone access.
- *Simulate SMS:* When toggled on, the simulator behaves as though the device's SMS functionality is available. This enables you to simulate design scenarios that require SMS access.
- *Simulate DB Read Structure:* When toggled on, the simulator takes the DB structure from the XML file that is specified in the Simulation 2 tab of the Options dialog[1669]. For relevant information, see the DB Read Structure[867] action.
- *Simulate Telephony:* When toggled on, the simulator behaves as though the device's telephony functionality is available. This enables you to simulate design scenarios that require telephone access.
- *Simulate In-App Purchases:* When selected, enables the simulation of In-App Purchases[1502] by using sample data that is stored in an XML file[1516]. The XML file to use is specified in the Simulation 2 tab of the Options dialog[1669].
- *System Theme Light/Dark:* Switches to the selected theme (light or dark).
- *Show Tab Order:* If tab ordering[1633] has been set, then select this option to show all tabbed controls with their respective tab-order number.
- *Log Errors Only:* Select this option to log errors only and to ignore other types of messages.
- *Set Default Options:* Resets Simulation pane options to their default settings[1663].

*Page Sources pane: options and features*

The following options are available in the Page Sources toolbar.

- *Evaluate XPath:* Opens the XPath/XQuery Window[1244], in which you can evaluate XPath expressions. XPath expressions can also be evaluated from the Styles & Properties Pane[274] while the simulation is running.
- *Clear Persistent Data and Restart:* Clears persistent data and restarts the simulation.
- *Restart Simulation:* Restarts the simulation at any time.

- *Search:* Enables searches for text in the page sources in the Page Sources pane, starting from the root element of page sources.
- In the Page Sources pane you can use copy-paste to copy parts of the tree to other locations in the tree. This is useful if you wish to copy data, such as DB records, in order to add more data for the simulation. The copied nodes are available only for the duration of the simulation.
- You can right-click a node and then insert, append, or add a child element or attribute. You can also rename an element or attribute by double-clicking its name. If the page structure is modified in any of these ways, then the page source is reloaded with the new structure and the updates will be correctly reflected in the displayed page.
- If you right-click a node of a page source and select **Load XML**, then the entire page source is replaced by the XML file you select. Note that, if the structure of the loaded XML file does not match the structure expected of this page source, then those parts of the design that are based on this page source will contain errors.
- If you right-click a node of a page source and select **Save XML**, then the entire page source is saved to the selected XML file.
- If you right-click a node of a page source, you can copy the XPath locator expression of that node to the clipboard by selecting the **Copy XPath** context menu command.

## Editing the XML trees of page sources in the Simulator

The XML trees in the simulator display the XML data of the various page sources and how these values change as the simulation progresses. You can edit the XML trees directly in the Simulator using cut/copy/past/delete and drag-and-drop, and you can also add, insert, and append attributes and elements to the tree. The editing commands are available in the context menu of the XML tree. The Simulation pane will instantly show the modified data. Modifying the XML tree in this way enables you to test the solution with modified XML data structures containing different manually entered data. As a result, you can quickly try out alternatives that contain different data and/or structures.

The context menu of XML trees in the simulator offers the following features:

- **Add / Insert / Append Attribute/Element**: The attribute or element is, respectively, added, inserted before, or appended after the item on which the context menu was called.
- **Load XML**: Loads an external XML file (that has the same structure and elements as the XML tree) into the XML tree.
- **Save XML**: Saves the structure and data of an XML tree to any location you like.
- **View in XMLSpy**: Opens the XML tree in Altova's XMLSpy program.
- **Overwrite $XML structure based on this tree**: Overwrites the structure of a page source with the structure of the XML tree in the simulator.

# 18.2 Simulation on Server

A server simulation uses MobileTogether Server to run the simulation (**Run | Use Server for Workflow Simulation**). It reports client messages in the Messages Pane 1385. Additional to testing that the design is error-free, this simulation enables you to test whether data sources are correctly located, whether URLs are correct, whether current server settings are appropriate, and whether the server has all permissions to access the used DBs, URLs, and files.

Simulating the workflow on the server works as follows:

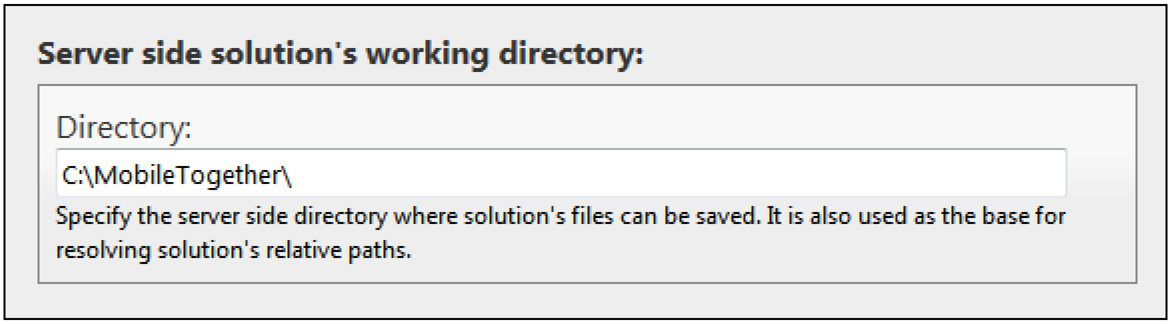
1. The workflow of the active design file in MobileTogether Designer is temporarily passed to MobileTogether Server. So the design file does not need to be deployed to the server in order to see how the design will work from the server.
2. The server serves the workflow to the simulator of MobileTogether Designer. In this way the simulator plays the role of a client.

## Simulation language

The simulation language for designer 1356 and server 1362 simulations is selected via the **Project | Simulation Language** 1606 command. The language of client simulations 1370 is the same as the language of the client mobile device on which the simulation runs.

## Running the simulation

1. Start MobileTogether Server. See the MobileTogether Server user manual for information about how to do this.
2. In the Web UI of MobileTogether Server, you must set the solution's working directory (**Settings | Server Side Solution's Working Directory**, *see screenshot below*). All relative paths in the design will be resolved relative to the directory specified in this setting. In order for server simulation to work correctly, enter the path of the directory where your referenced files are saved.



**Server side solution's working directory:**

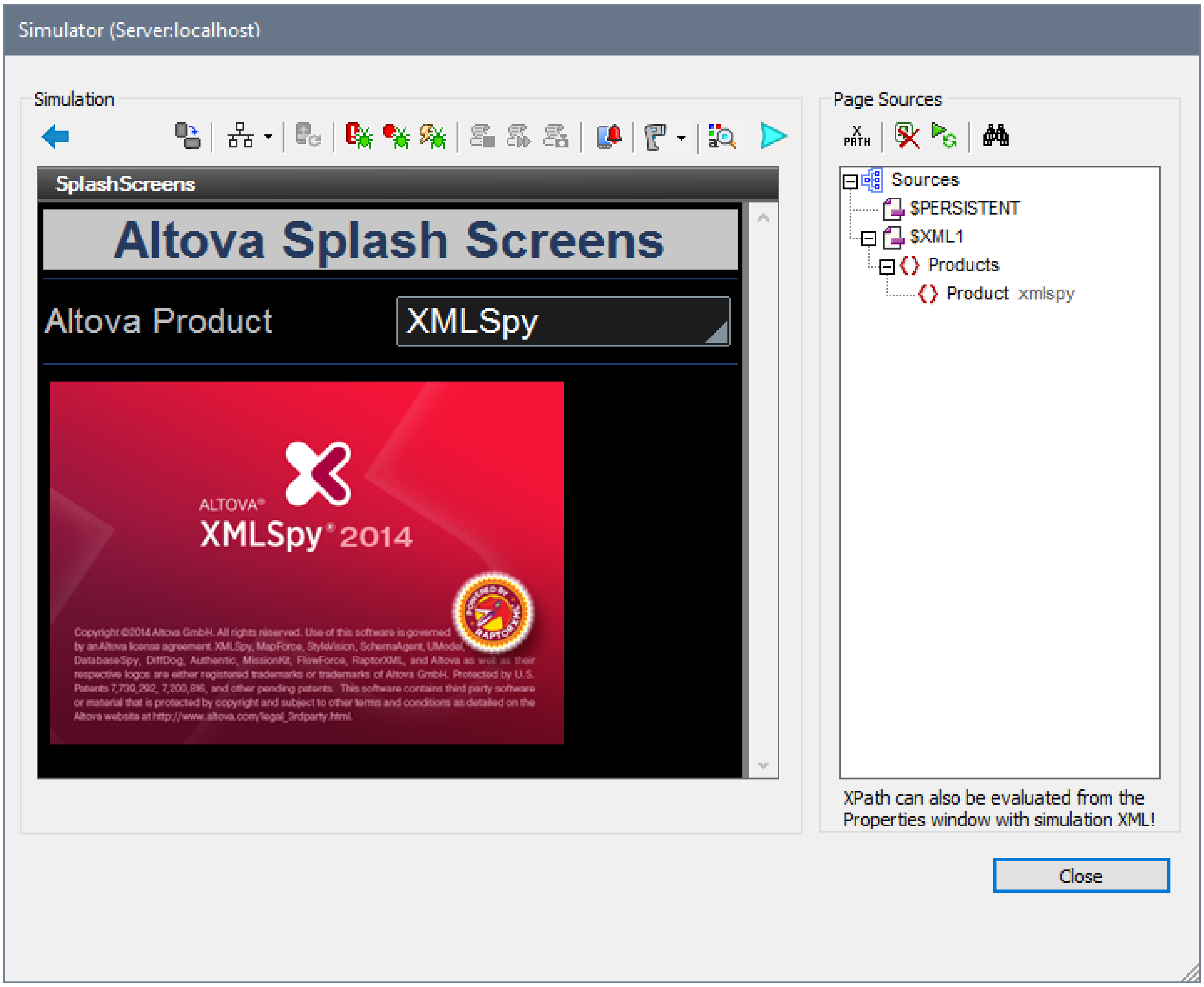
Directory:

C:\MobileTogether\

Specify the server side directory where solution's files can be saved. It is also used as the base for resolving solution's relative paths.

3. In MobileTogether Designer, make sure that the server settings 1663 are correctly defined.
4. In MobileTogether Designer, select **Run | Use Server for Workflow Simulation**.
5. If you are prompted for credentials to access the server, you can enter the `user-name/password` combination of `root/root`, or any other user credentials that have been set up with the privilege of running server simulations. See the MobileTogether Server user manual for information about assigning privileges to users.

The simulator window is opened and runs the workflow.



XPath can also be evaluated from the Properties window with simulation XML!

## File locations

If MobileTogether Server is used for simulation, files referenced by the design must be located either directly in the directory designated as the *Server Side Solution's Working Directory*, or in a descendant directory of this directory. (The Working Directory setting is made in the MobileTogether Server settings page.)

- If absolute paths are used, the file must be located in the Working Directory or a descendant directory of the Working Directory.
- If relative paths are used, the path is resolved relative to the Working Directory.

## Simulator features

The simulator window provides the following features:

- The left-hand (Simulation) pane displays the simulation. Options for the Simulation pane are described below.

- The right-hand (Page Sources) pane shows the changes taking place in the XML data as the simulation progresses. Options for the Page Sources pane are described below.
- While the simulation progresses, the Messages Pane[1385] of the MobileTogether Designer GUI provides a detailed and step-by-step report of all the activities taking place. You can therefore see what is happening at each step in the progress of the workflow. This is an invaluable feature for testing and debugging design files.
  - *Simulations on the designer report activities on both server and client.*
  - *Simulations on the server report client messages.*
  - *Simulations on the client report server messages*.
- Controls that require user interaction are enabled. In the screenshot above, for example, the combo box is enabled.

*Simulation pane toolbar*

The toolbar of the Simulation pane contains the following buttons (from left to right):

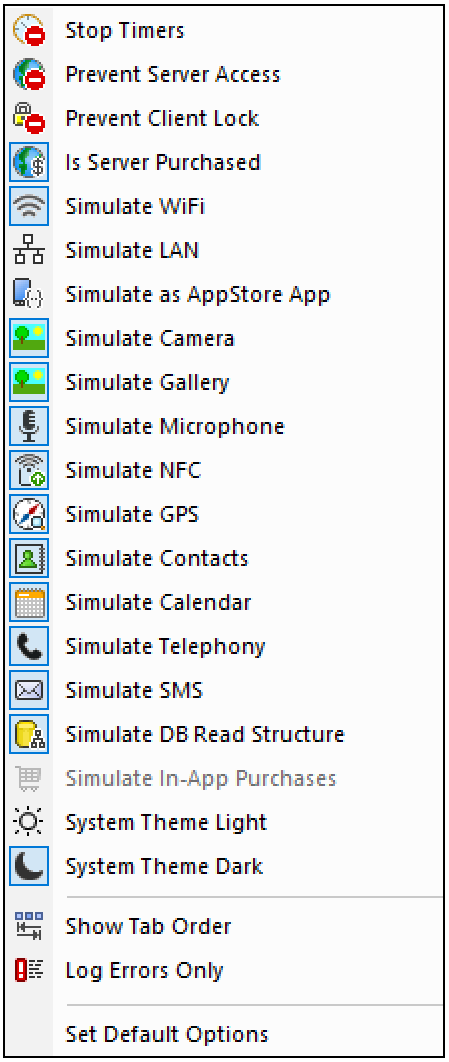
- *Back:* If the page is a subpage[257], then clicking **Back** closes the subpage. If the page is a top page[257], the simulator closes. Also see **`OnBackButtonClicked`**[393].
- *Change Orientation:* You can switch the view between landscape and portrait.
- *Simulator Options:* Drops down a menu of simulator options (described below under *Simulator options*).
- *Simulate Return to App (Reopen):* Enabled when page refreshes have been defined to occur on page reopens[390]. Refreshes the page during simulations.
- *Stop at Next Error:* Stops the simulation at the next XPath error and displays the XPath expression in the XPath Debugger[1388].
- *Stop at Next Breakpoint:* Stops the simulation at the next breakpoint (which may be on an action or XPath expression) and opens the appropriate Debugger[1388].
- *Stop at Next Action:* Stops the simulation at the action/s of the next event that is triggered and displays the action/s in the Actions Debugger[1390].
- *Stop Recording Test Case:* When the recording of a test case[1401] is started, the simulator is opened and the recording runs in the simulator. Clicking this button: (i) stops the recording of the test case[1401], and (ii) opens the Recorded Test Case Confirmation dialog, in which you specify the name of the recorded test case.
- *Playback Next Step:* Plays back the next step in the currently running test-case playback[1403]. This button is enabled when playback is set to step-by-step. See the *Playback Options* section of Playing Back a Test Case[1403].
- *Record Snapshot:* Records a snapshot of the test case that is being recorded[1401]. This button is enabled when the recording of snapshots has been set to manual (not automatic). See the *Recording Options* section of Recording a Test Case[1401].
- *Push Notifications:* If the design contains a push notification, opens the Manage Recorded Push Notification (Simulation) dialog. See Simulating Push Notifications[1133] for a description of usage.
- *Barcode Scanner:* Drops down a menu that has three parts, one for each supported barcode scanner[1145]. For each scanner, you can simulate receiving barcode data from the scanner. In the case of Zebra scanners, you can also simulate connection established, receiving image data, and connection terminated. Note that (for Zebra scanners) you must simulate a connection-establishment before the options for subsequent simulations become available. For some of these simulations, a dialog will appear in which you must supply the data for the simulation (for example an ISBN or an image file).
- *Styles Inspector:* Open the Styles Inspector window[1332].
- *Purchase:* If the design enables in-app purchases, provides a list of all the in-app purchase products that are available for purchase and enables you to purchase one or more. The list of products is obtained from the In-app Purchase XML data file for simulations[1669]. See In-App Purchases[1502] for more information.
- *Purchased:* If the design enables in-app purchases, provides a list of all the in-app purchase products

that are listed as purchased (that is, as a **`Purchase`** element) in the In-app Purchase XML data file for simulations[1669]. If you select one of these purchased products, then this product together with all its purchase properties is considered, in the simulation, to have been purchased. If you were to subsequently run the Query Purchases[934] action or Restore Purchases[934] action, then the products that you selected will be returned (by the action) as having been purchased. See In-App Purchases[1502] for more information.

- *Simulate Reception of MQTT Message:* If the design enables the receipt of MQTT messages, simulates actions that are performed on receiving an MQTT message. The MQTT messages to simulate are taken from a sample file specified in the Options dialog[1669].
- *Simulate Reception of Broadcast Message:* If the design enables the receipt of Broadcast messages, simulates actions that are performed on receiving a broadcast message. The broadcast messages to simulate are taken from a sample file specified in the Options dialog[1669].
- *Submit:* If the page is not the last page, then clicking **Submit** takes you to the next page. If the page is the last page, then the workflow is exited. Also see **`OnSubmitButtonClicked`**[394].
- *Refresh Page:* If the Page Refresh option[390] has been set to *Manual*, then this button becomes visible. On clicking **Refresh Page**, the page is updated with the changed data.

*Simulator options*

On clicking the **Simulator Options** toolbar button, a dropdown list of options for the simulator is displayed *(screenshot below)*. Each option is described below. Default settings are specified in the Simulation 1[1668] and Simulation 2[1669] tabs of the Options dialog. Note that, if you change a setting in the simulator, the new setting becomes the default and will be shown as such in the Simulation 1[1668] and Simulation 2[1669] tabs of the Options dialog.



- *Stop Timers:* If a timer has been set to run at intervals 390 and actions have been defined to be executed at these intervals, you can stop timers (and, consequently, actions) by clicking **Stop Timers**. This will clear up the clutter of messages generated by these actions, and will allow you to more easily analyze other messages and aspects of the workflow.
- *Prevent Server Access:* When selected, disables access to the server, and so allows you to test the solution's behavior in a server-connection-error scenario. When unselected, server access is allowed. For more information about this feature, see Server Connection Errors 394 .
- *Prevent Client Lock:* When selected, prevents the client being locked from server access when the Lock Client 919 action is executed. If the server is inaccessible because the Lock Client 919 action has been executed from another client, then preventing the lock will of course not work.
- *Is Server Purchased:* For simulations in the designer and for trial runs on client, simulates that MobileTogether Server licenses have been purchased. For simulations on the server, the actual purchase-state of licenses on the server is returned.
- *Simulate WiFi:* Sets the `mt-connected-via-wifi` 1262 XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This allows the simulator to behave as though WiFi access is available. In this way, you can simulate design scenarios in which WiFi access is required.
- *Simulate LAN:* Sets the `mt-connected-via-lan` 1262 XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This allows the simulator to behave as though a LAN connection

is available. As a result, you can simulate design scenarios that require a LAN connection.

- *Simulate as AppStore App:* Sets the static global variable **`MT_IsAppStoreApp`** 1300 to `true()` when enabled, to `false()` when disabled. This allows simulations to be carried out that are conditional on the value of this variable.
- *Simulate Camera:* When toggled on, the simulator behaves as though the device's camera is available. This enables you to simulate design scenarios that require camera access.
- *Simulate Gallery:* When toggled on, the simulator behaves as though the device's photo gallery is available. This enables you to simulate design scenarios that require gallery access.
- *Simulate Microphone:* When toggled on, the simulator behaves as though the device's microphone is available. This enables you to simulate design scenarios that require microphone access.
- *Simulate NFC:* When selected, enables NFC functionality so that NFC actions can be executed. Actual NFC data is supplied to the simulator via NFC sample files 1377.
- *Simulate Bluetooth (BT):* When toggled on, the simulator behaves as though the device's Bluetooth connectivity is available. This enables you to simulate Barcode Scanner scenarios 1145 that require Bluetooth.
- *Simulate GPS:* When selected, enables geolocation functionality so that geolocation features can be tested. Dummy geolocations can be supplied via the Geolocations XML file 1372, which is used specifically to supply geolocations for simulations.
- *Simulate Contacts:* When toggled on, the simulator behaves as though the device's address book is available. This enables you to simulate design scenarios that require access to the address book. The address book is simulated either from a sample file 1381 or from your Microsoft Outlook contacts 1663. Which option to use is specified in the Simulation 2 tab of the Options dialog 1669.
- *Simulate Calendar:* When toggled on, the simulator behaves as though the device's calendar is available. This enables you to simulate design scenarios that require access to the calendar. The calendar is simulated either from a sample file 1382 or from your Microsoft Outlook calendar 1663. Which option to use is specified in the Simulation 2 tab of the Options dialog 1669.
- *Simulate Telephony:* When toggled on, the simulator behaves as though the device's telephony functionality is available. This enables you to simulate design scenarios that require telephone access.
- *Simulate SMS:* When toggled on, the simulator behaves as though the device's SMS functionality is available. This enables you to simulate design scenarios that require SMS access.
- *Simulate DB Read Structure:* When toggled on, the simulator takes the DB structure from the XML file that is specified in the Simulation 2 tab of the Options dialog 1669. For relevant information, see the DB Read Structure 867 action.
- *Simulate Telephony:* When toggled on, the simulator behaves as though the device's telephony functionality is available. This enables you to simulate design scenarios that require telephone access.
- *Simulate In-App Purchases:* When selected, enables the simulation of In-App Purchases 1502 by using sample data that is stored in an XML file 1516. The XML file to use is specified in the Simulation 2 tab of the Options dialog 1669.
- *System Theme Light/Dark:* Switches to the selected theme (light or dark).
- *Show Tab Order:* If tab ordering 1633 has been set, then select this option to show all tabbed controls with their respective tab-order number.
- *Log Errors Only:* Select this option to log errors only and to ignore other types of messages.
- *Set Default Options:* Resets Simulation pane options to their default settings 1663.

*Page Sources pane: options and features*
The following options are available in the Page Sources toolbar.

- *Evaluate XPath:* Opens the XPath/XQuery Window 1244, in which you can evaluate XPath expressions. XPath expressions can also be evaluated from the Styles & Properties Pane 274 while the simulation is running.
- *Clear Persistent Data and Restart:* Clears persistent data and restarts the simulation.
- *Restart Simulation:* Restarts the simulation at any time.

- *Search:* Enables searches for text in the page sources in the Page Sources pane, starting from the root element of page sources.
- In the Page Sources pane you can use copy-paste to copy parts of the tree to other locations in the tree. This is useful if you wish to copy data, such as DB records, in order to add more data for the simulation. The copied nodes are available only for the duration of the simulation.
- You can right-click a node and then insert, append, or add a child element or attribute. You can also rename an element or attribute by double-clicking its name. If the page structure is modified in any of these ways, then the page source is reloaded with the new structure and the updates will be correctly reflected in the displayed page.
- If you right-click a node of a page source and select **Load XML**, then the entire page source is replaced by the XML file you select. Note that, if the structure of the loaded XML file does not match the structure expected of this page source, then those parts of the design that are based on this page source will contain errors.
- If you right-click a node of a page source and select **Save XML**, then the entire page source is saved to the selected XML file.
- If you right-click a node of a page source, you can copy the XPath locator expression of that node to the clipboard by selecting the **Copy XPath** context menu command.

**Note:** If you have problems connecting to the server, try checking the settings on the server. See the MobileTogether Server user manual for information about how to do this.

## Editing the XML trees of page sources in the Simulator

The XML trees in the simulator display the XML data of the various page sources and how these values change as the simulation progresses. You can edit the XML trees directly in the Simulator using cut/copy/past/delete and drag-and-drop, and you can also add, insert, and append attributes and elements to the tree. The editing commands are available in the context menu of the XML tree. The Simulation pane will instantly show the modified data. Modifying the XML tree in this way enables you to test the solution with modified XML data structures containing different manually entered data. As a result, you can quickly try out alternatives that contain different data and/or structures.

The context menu of XML trees in the simulator offers the following features:

- **Add / Insert / Append Attribute/Element**: The attribute or element is, respectively, added, inserted before, or appended after the item on which the context menu was called.
- **Load XML**: Loads an external XML file (that has the same structure and elements as the XML tree) into the XML tree.
- **Save XML**: Saves the structure and data of an XML tree to any location you like.
- **View in XMLSpy**: Opens the XML tree in Altova's XMLSpy program.
- **Overwrite $XML structure based on this tree**: Overwrites the structure of a page source with the structure of the XML tree in the simulator.

### Server IP address and network firewall settings

Your server can have a public IP address (accessible over the Internet) and/or a private IP address (accessible within a private network; for example, via WiFi within a company network). If a mobile client device tries to connect via the Internet using the server's private IP address, then the connection will not work. This is because the private IP address is not known on the Internet and cannot be resolved. If a client device uses a private IP address, then the client device must already have access to the private network.

To ensure that the server can be accessed, do one of the following:

- Provide the server with a public IP address so that it can be reached via the Internet. On the client device, use this public IP address to access the server.
- If you use a firewall and install MobileTogether Server on a server with a private IP address (inside the private network), then use the network firewall to forward requests sent to a public IP-address/port-combination to your MobileTogether Server server. On the client device, use the public IP address.

You must also ensure that the firewall is configured to allow access to the server port used for MobileTogether Client communication. The ports used by MobileTogether Server are specified in the Settings page of the Web UI of MobileTogether Server (*see the MobileTogether Server user manual*). On the client device, this is the port that must be specified as the server port to access.

**Tip:** Port 80 is usually open on most firewalls by default. So, if you are having difficulties with firewall settings and if port 80 is not already bound to some other service, you could specify port 80 as the MobileTogether Server port for client communication.

# 18.3 Simulation on Client

This simulation method runs the workflow on your mobile device by using MobileTogether Designer as a server. It reports server messages in the Messages Pane[1385] of MobileTogether Designer. Simulation on the client assumes that your mobile device can connect to your PC over wireless LAN. To run a client simulation, carry out the steps listed below. (For client simulations of compiled apps, see the box below.)

1. Client: Set up a new server
    1. Start MobileTogether on your client device and click the MobileTogether app's **Settings | Server** icon.
    2. Add a new server and enter the following data: a connection name, the IP address of the machine on which MobileTogether Designer is running, and the port number on which MobileTogether Designer is listening (by default this is 8083). (In MobileTogether Designer, click **Run | Trial Run on Client**[1618] to check at which port MobileTogether Designer will be listening.)
    3. Save the server settings.

2. PC: Start the simulation
    1. If MobileTogether Server is running on your PC and uses the same port as the designer's listening port (*see previous point*), then stop MobileTogether Server running as a service.
    2. In MobileTogether Designer, you can change MobileTogether Designer's listening port. Do this in the *Trial Run on Client* tab of the Options dialog (**Tools | Options**).
    3. In MobileTogether Designer, select **Run | Trial Run on Client**. This does two things: (i) opens, in designer, a Trial Run on Client dialog that will show the page sources of the solution, and (ii) opens the designer's connection to the client. Note that communication between designer and client starts only when Trial Run on Client has been started in MobileTogether Designer. In the Trial Run on Client dialog, click **Find** to search for text within the returned data.
    4. On the client device, refresh the view to see a listing of the designs (the design list) that are currently open in MobileTogether Designer.

3. Client: Run the simulation
    1. On the client, select from the design list the solution that you want to test.
    2. This opens a prompt in MobileTogether Designer asking if you want to start the solution. Click **Yes**. The trial run of the solution starts on the client and the solution's page sources now appear in the dialog box of MobileTogether Designer.
    3. Click **Back/Return** to stop the current solution. A prompt appears asking if you want to stop. Click **Yes**.

**Note:** A design file can be used by only one client at a time for client simulation.

## Client simulations of compiled apps

The procedure for simulating a trial run of a compiled app[1471] is similar to that for solutions described above. The main difference is that the designer's connection details will have already been specified in the compiled app (*see Screen 3 of the Code-Generation wizard*[1472])[1472]. As a result, these details do not need to be set on the client.

Additionally, after you have compiled an app-store app, the design can be changed as often as you like. You will not need to recompile the app to run a client simulation. The app will connect to MobileTogether Designer and use the currently open version of the design.

The steps to set up client simulations of a compiled app are as follows:

1. In MobileTogether Designer, go to the Trial Run on Client tab of the Options dialog 1667 and make sure that the listening port specified there is the same as that you specified in the compiled app (*in Screen 3 of the Code-Generation wizard* 1472).
2. In MobileTogether Designer, open the project file (`.mtd`) of the compiled app that you want to test, and edit it as required.
3. Start the compiled app on your device. The compiled app will use the project file in MobileTogether Designer (including the last edit even if this was not saved), and the pages sources and simulation messages will be displayed in MobileTogether Designer.

## Simulation language

The simulation language for designer 1356 and server 1362 simulations is selected via the **Project | Simulation Language** 1606 command. The language of client simulations 1370 is the same as the language of the client mobile device on which the simulation runs.

## XML trees of page sources in the Simulator

The XML trees in the simulator display the XML data of the various page sources and how these values change as the simulation progresses. The context menu of XML trees in the simulator offers the following features:

- **Copy XPath**: Copies to the clipboard an XPath locator expression that locates the selected tree node.
- **Save XML**: Saves the structure and data of an XML tree to any location you like.
- **View in XMLSpy**: Opens the XML tree in Altova's XMLSpy program.
- **Overwrite $XML structure based on this tree**: Overwrites the structure of a page source with the structure of the XML tree in the simulator.
- **Send XML to Designer**: Sends all client XML data to the designer, enabling you to analyze the solution's XML data.

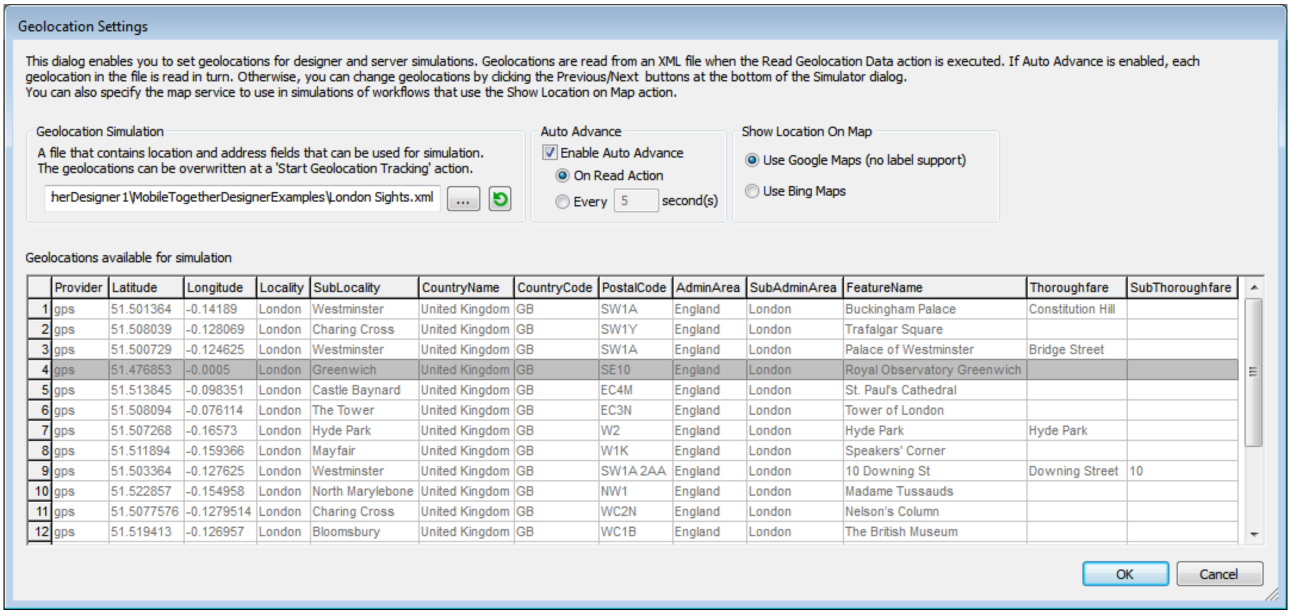
# 18.4 Geolocation Settings

The Geolocation Settings dialog (*screenshot below*) enables you to specify geolocations from an XML file for designer 1356 and server 1362 simulations. You need to specify geolocations in this way because these simulations do not use a mobile device and therefore have no geolocation data available to them. These geolocations stand in for the actual geolocations of a mobile device.

The Geolocation Settings dialog is accessed via the Simulator dialog:

1. Click **Run | Simulate Workflow (F5)** or **Run | Use Server for Workflow Simulation (Ctrl+F5)** to access the Simulator dialog.
2. In the Simulator dialog, click the **Geolocation** button at bottom left to open the Geolocation Settings dialog (*screenshot below*). The settings made here will be used for both designer 1356 and server 1362 simulations.

**Note:** If no geolocation action is used in the design, the **Geolocation** button will be **not be displayed** in the Simulator dialog.



Geolocation Settings

This dialog enables you to set geolocations for designer and server simulations. Geolocations are read from an XML file when the Read Geolocation Data action is executed. If Auto Advance is enabled, each geolocation in the file is read in turn. Otherwise, you can change geolocations by clicking the Previous/Next buttons at the bottom of the Simulator dialog.
You can also specify the map service to use in simulations of workflows that use the Show Location on Map action.

Geolocation Simulation

A file that contains location and address fields that can be used for simulation.
The geolocations can be overwritten at a 'Start Geolocation Tracking' action.

herDesigner1\MobileTogetherDesignerExamples\London Sights.xml

Auto Advance

☑ Enable Auto Advance

● On Read Action

○ Every 5 second(s)

Show Location On Map

● Use Google Maps (no label support)

○ Use Bing Maps

Geolocations available for simulation

| | Provider | Latitude | Longitude | Locality | SubLocality | CountryName | CountryCode | PostalCode | AdminArea | SubAdminArea | FeatureName | Thoroughfare | SubThoroughfare |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | gps | 51.501364 | -0.14189 | London | Westminster | United Kingdom | GB | SW1A | England | London | Buckingham Palace | Constitution Hill | |
| 2 | gps | 51.508039 | -0.128069 | London | Charing Cross | United Kingdom | GB | SW1Y | England | London | Trafalgar Square | | |
| 3 | gps | 51.500729 | -0.124625 | London | Westminster | United Kingdom | GB | SW1A | England | London | Palace of Westminster | Bridge Street | |
| 4 | gps | 51.476853 | -0.0005 | London | Greenwich | United Kingdom | GB | SE10 | England | London | Royal Observatory Greenwich | | |
| 5 | gps | 51.513845 | -0.098351 | London | Castle Baynard | United Kingdom | GB | EC4M | England | London | St. Paul's Cathedral | | |
| 6 | gps | 51.508094 | -0.076114 | London | The Tower | United Kingdom | GB | EC3N | England | London | Tower of London | | |
| 7 | gps | 51.507268 | -0.16573 | London | Hyde Park | United Kingdom | GB | W2 | England | London | Hyde Park | Hyde Park | |
| 8 | gps | 51.511894 | -0.159366 | London | Mayfair | United Kingdom | GB | W1K | England | London | Speakers' Corner | | |
| 9 | gps | 51.503364 | -0.127625 | London | Westminster | United Kingdom | GB | SW1A 2AA | England | London | 10 Downing St | Downing Street | 10 |
| 10 | gps | 51.522857 | -0.154958 | London | North Marylebone | United Kingdom | GB | NW1 | England | London | Madame Tussauds | | |
| 11 | gps | 51.5077576 | -0.1279514 | London | Charing Cross | United Kingdom | GB | WC2N | England | London | Nelson's Column | | |
| 12 | gps | 51.519413 | -0.126957 | London | Bloomsbury | United Kingdom | GB | WC1B | England | London | The British Museum | | |

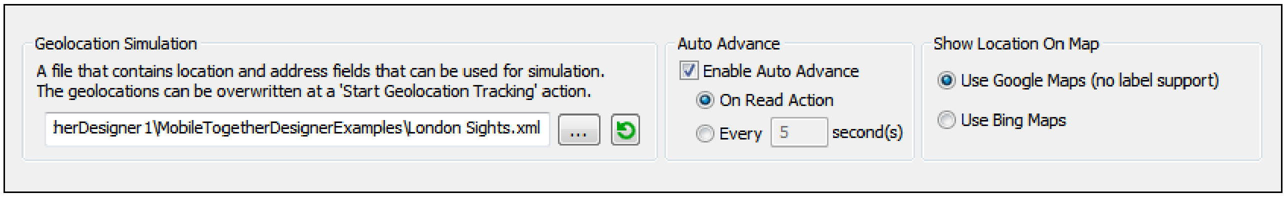
## The geolocation settings

You can make the following geolocation settings:

- *Geolocation XML file:* Contains the default geolocations to use for simulations. The XML file must have the structure listed below. The **Browse** button enables you to browse for the file. The **Refresh** button enters the geolocation data of the XML file into the available geolocation values pane in the lower part of the dialog. The default file can be overridden by a geolocations simulation file that is specified in the Start Geolocation Tracking 735 action.
- *Auto Advance:* If Auto Advance is enabled, each geolocation in the XML file is read in turn during simulation. You can specify the time interval between the reading of each geolocation. If Auto Advance

is not specified, you can change geolocations during simulation by clicking the **Previous** or **Next** buttons at the bottom left of the Simulator dialog. Note that these geolocation values run in the simulator only. They are passed to page source tree nodes (including the `$GEOLOCTION` tree) only when such an action is explicitly specified in the design.

- *Show Location on Map:* Selects which map application to open in the web browser when the Show Geolocation on Map action is executed.



Geolocation Simulation

A file that contains location and address fields that can be used for simulation. The geolocations can be overwritten at a 'Start Geolocation Tracking' action.

herDesigner 1\MobileTogetherDesignerExamples\London Sights.xml

Auto Advance

☑ Enable Auto Advance

● On Read Action

○ Every 5 second(s)

Show Location On Map

● Use Google Maps (no label support)

○ Use Bing Maps

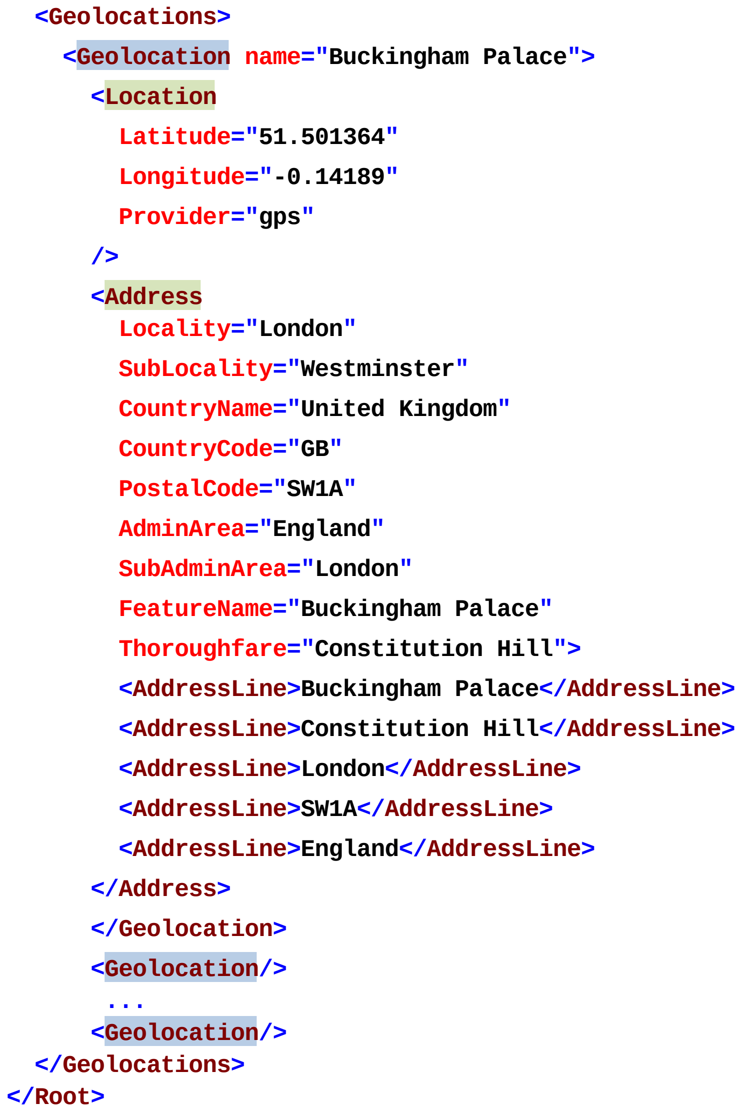
## The geolocation XML file structure

In order for MobileTogether Designer to correctly read geolocation data, the geolocation XML file must have a structure similar to that shown in the listing below. Attributes may be omitted or their values may be the empty string. However, the `//Location/Latitude` and `//Location/Longitude` attributes need to be present

### ⊟ *Example structure of the Geolocation XML file*

This example shows one `Geolocation` element expanded. Not all attributes of the **`Location`** and **`Address`** elements are used. For a complete list of attributes, see the next listing. For the lexical format of latitude and longitude values, see the section *Geolocation Input String Formats* below.

```
<Root>
```



```
    <Geolocations>
      <Geolocation name="Buckingham Palace">
        <Location
          Latitude="51.501364"
          Longitude="-0.14189"
          Provider="gps"
        />
        <Address
          Locality="London"
          SubLocality="Westminster"
          CountryName="United Kingdom"
          CountryCode="GB"
          PostalCode="SW1A"
          AdminArea="England"
          SubAdminArea="London"
          FeatureName="Buckingham Palace"
          Thoroughfare="Constitution Hill">
          <AddressLine>Buckingham Palace</AddressLine>
          <AddressLine>Constitution Hill</AddressLine>
          <AddressLine>London</AddressLine>
          <AddressLine>SW1A</AddressLine>
          <AddressLine>England</AddressLine>
        </Address>
        </Geolocation>
        <Geolocation/>
         ...
        <Geolocation/>
    </Geolocations>
  </Root>
```

*All attributes of the Geolocation XML file*

```
<Root>
  <Geolocations>
    <Geolocation name="">
      <Location
        AccuracyHorizontal=""
        AccuracyVertical=""
        Altitude=""
        Latitude=""
        Longitude=""
        MagneticHeading=""
        Provider=""
        Speed=""
        Time=""
      />
      <Address
        AdminArea=""
        CountryCode=""
        CountryName=""
        FeatureName=""
        Locality=""
        Phone=""
        PostalCode=""
        Premises=""
        SubAdminArea=""
        SubLocality=""
        SubThoroughfare=""
        Thoroughfare=""
        Url="">
        <AddressLine/>
         ... AddressLine* elements ...
        <AddressLine/>
    </Geolocation>
  </Geolocations>
</Root>
```

*Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (`"`) while unit indicators that are escaped are highlighted in blue (`""`).

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example*: **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example*: **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example*: **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example*: **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`

*Example*: **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example*: **`33.33  -22.22`**

*Examples of format-combinations:*

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

## 18.5 NFC Sample Files

If you wish to simulate the discovery of an NFC tag, you must use an NFC sample file in place of the NFC tag. (This is because your desktop machine is not enabled for NFC.) Select the NFC sample file by clicking the **NFC Samples** button at the bottom of the simulation pane and then browsing for the NFC samples file. The NFC samples file must have the structure shown in the listing below.

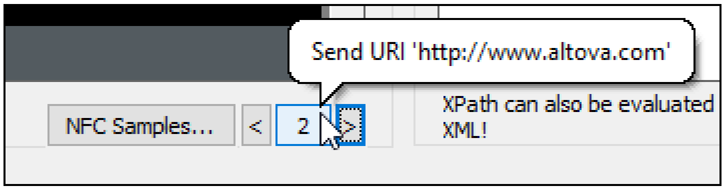
*Template for NFC sample file*

```
<Root>
   <NFCS>
      <NFC name="Text" tooltip="sends a well known text 'This is my text'">
<Root>
   <Tag Id=""/>
   <NdefMessage
      CanMakeReadOnly=""
      IsWriteable=""
      MaxSize=""
      Type="">
      <NdefRecord
         Id=""
         TypeNameField=""
         RecordTypeDefinition=""
         Type=""
         Text=""
         Language=""
         URI=""
         Payload=""
         MimeType=""
         ExternalDomain=""
         ExternalPackageName="">
         <NdefRecord />
      </NdefRecord>
      <NdefRecord />
         ...
      <NdefRecord />
   </NdefMessage>
</Root>
      </NFC>
      <NFC/>
       ...
      <NFC/>
   </NFCS>
</Root>
```

As you can see from the listing above, each **NFC** element corresponds to a single message (**NdefMessage** element), which contains multiple records (**NdefRecord** elements).

Note that each sample file can have multiple messages. Once an NFC sample file has been selected for use in a simulation, the index of the currently selected *message* is shown in the simulator (*see screenshot below*). Furthermore, if you place the mouse cursor over the message number, a tooltip is displayed (*see screenshot*).

This tooltip is the value of that message's **`NFC/@tooltip`**, attribute. To set a new message as the current message, use the Next and Previous buttons (*see screenshot*).



## Reading message data from an NFC sample file

To read data from a particular message into the `$MT_NFC` tree from the NFC sample file, do the following:

1. Start NFC (by triggering the Start NFC[746] action).
2. Specify the NFC sample file by clicking **NFC Samples** and browsing for the file.
3. Make sure that the number of the message you want to read from the NFC sample file is the currently displayed message number (*in the screenshot above, the number 2*).
4. Click this message number. The message data in the file will be read and loaded into the `$MT_NFC` tree.

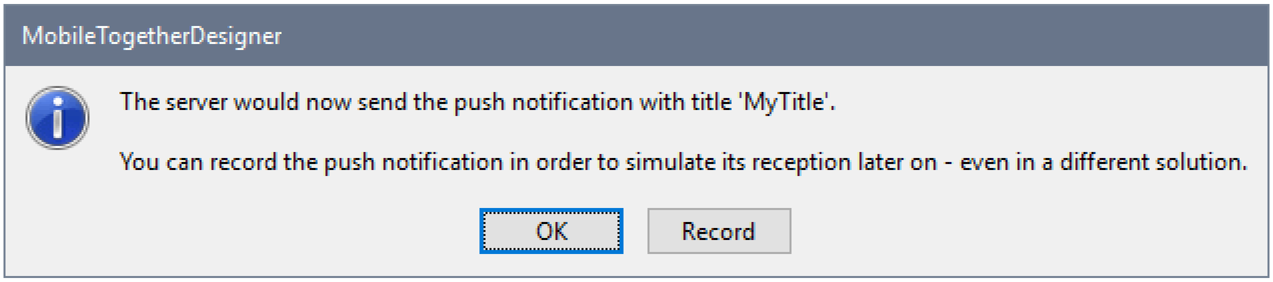
# 18.6 Push Notification Simulations

A push notification (PN) contains data related to: (i) the PN's short message, (ii) the PN's big message, and (iii) the PN's payload. If the sending and receiving solutions are one and the same solution, then, in a simulation, the data transfer between sending and receiving parts is carried out within that one solution's simulation; the simulation in this case is straightforward.

However, if the sending solution and receiving solution are different, then the simulation mechanism is as follows: PN data sent during a simulation of the sending solution is recorded in a MT PN Simulation file (which has a `.mtpnsim` extension). When the receiving solution is simulated, you can load that `.mtpnsim` file. The simulator will now display all the sets of PN data in the `.mtpnsim` file, and you can select the PN that you want to simulate.

## Recording PN Simulation Data

If, while simulating a sending solution, you trigger an event that sends a PN, then the dialog shown below appears.


MobileTogetherDesigner

The server would now send the push notification with title 'MyTitle'.

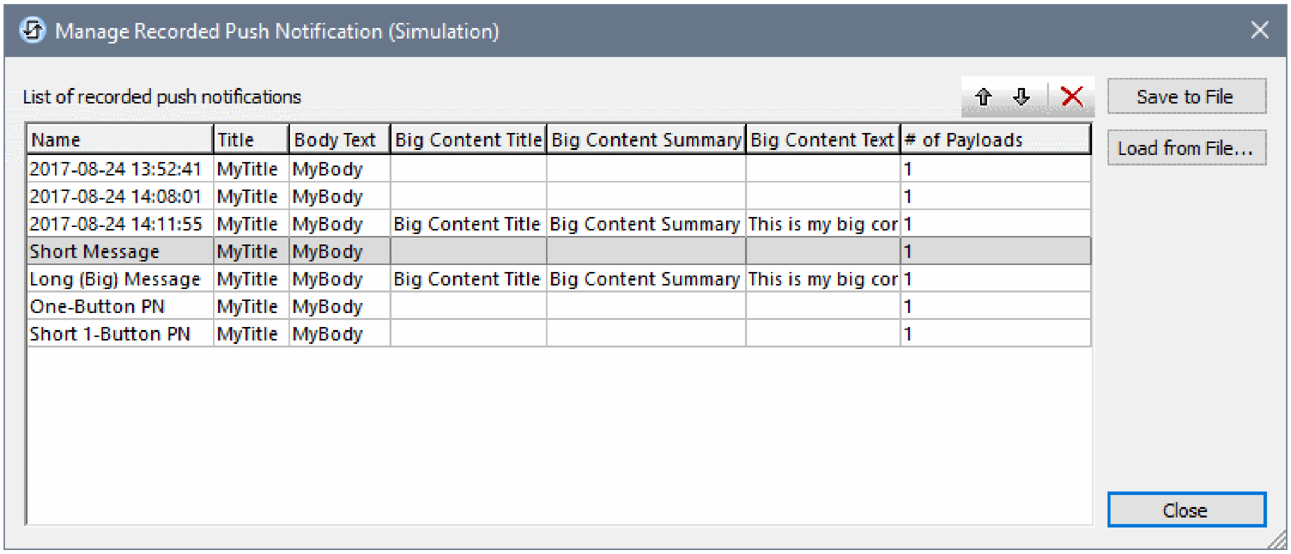
You can record the push notification in order to simulate its reception later on - even in a different solution.

Click **Record** to record the PN data to memory. You can record multiple PNs to memory in this way. On closing the simulation, the recorded PNs are in memory—but not yet saved to file.

When you save or close the solution file, you will be informed that there is unsaved recorded PN data in memory, and you will be asked whether you wish to save this data to file. If you select **Yes**, then the different sets of recorded PN data in memory will be saved to a MT PN Simulation file in the same folder as the solution. This file will be named with the following pattern: `YourSolutionName.mtpnsim`. Any additional PNs sent during the current or subsequent simulations of that solution will be saved to the same file. Each set of PN data in the file is identified by a name, which is that PN's recording date and time.

## Loading recorded PN Simulation Data into the simulator

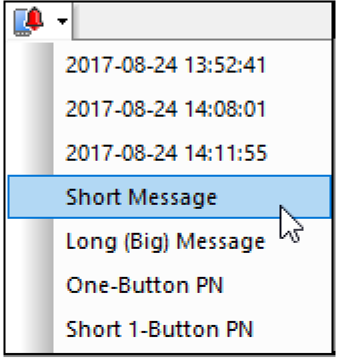
To load recorded PN simulation data from an MT PN Simulation file (`.mtpnsim` file), start a simulation of the receiving solution (an MTD file) and then click the **Push Notifications** icon 🔔▾ in the simulator's toolbar. This causes the Manage Recorded Push Notification (Simulation) dialog (*see screenshot below*) to appear. Click **Load from File**, then browse for the `.mtpnsim` file you want to load, and click **Open**. The recorded PN data in this file will be loaded into the dialog (*see screenshot*) and into memory. It will not automatically be saved to the MTD file. If you reload the MTD file without saving, then the recorded PN data will have to be reloaded from the `.mtpnsim` file. Click **Save to File** to save the recorded PN data to the MTD file; this will prevent you having to reload the `.mtpnsim` file. If you load PN data from another `.mtpnsim` file, then the new data will overwrite the PN data in memory. If you now wish to overwrite any recorded PN data in the MTD file, click **Save to File** in this dialog.


Manage Recorded Push Notification (Simulation)
List of recorded push notifications
Save to File
Load from File...
Close

| Name | Title | Body Text | Big Content Title | Big Content Summary | Big Content Text | # of Payloads |
|---|---|---|---|---|---|---|
| 2017-08-24 13:52:41 | MyTitle | MyBody | | | | 1 |
| 2017-08-24 14:08:01 | MyTitle | MyBody | | | | 1 |
| 2017-08-24 14:11:55 | MyTitle | MyBody | Big Content Title | Big Content Summary | This is my big cor | 1 |
| Short Message | MyTitle | MyBody | | | | 1 |
| Long (Big) Message | MyTitle | MyBody | Big Content Title | Big Content Summary | This is my big cor | 1 |
| One-Button PN | MyTitle | MyBody | | | | 1 |
| Short 1-Button PN | MyTitle | MyBody | | | | 1 |

In the Manage Recorded Push Notification (Simulation) dialog (*screenshot above*), each recorded PN is shown on a separate line, and is shown with its name, short message data, big message data, and payload information. You can change the order of the PNs by selecting one or more of them and clicking the **Move Up** and **Move Down** toolbar icons (located top right). You can delete a PN by selecting it and clicking the **Delete** toolbar icon. You can also edit the names of PNs so that you can identify them more easily; to do this, double-click the name and edit. Changes made in the dialog are made in memory. To save the changes to the MTD file, click **Save to File**.

To select which PN is used in the simulation, click the dropdown arrow of the **Push Notifications** icon in the simulator's toolbar. This displays a list of all the PNs that are currently in memory (*see screenshot below*). The order in which PNs are displayed is the same as their current order in the Manage Recorded Push Notification (Simulation) dialog (*compare screenshot below with that above*).



After you select a PN from this list, the solution will simulate that it has received the selected PN. To simulate the receipt of another PN, select a new PN from the dropdown list.

## 18.7 Contacts Sample Files

In order to simulate the address book of a device you can use a sample contacts file that has the XML structure shown in the listing below. You can then specify, in the Simulation pane of the Options dialog[1663], that this file is to be used for simulations involving the Read Contacts[692] action. Alternatively, you can use your Microsoft Outlook contacts for the simulation. The setting that specifies which of these two alternatives to use is in the Simulation pane of the Options dialog[1663].

*Template for sample address book*

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
   <Contact Id="">
      <Name Prefix="" First="" Middle="" Last="" Suffix=""/>
      <Image Image=""/>
      <Address Description="" Country="" PostalCode="" City="" Street=""/>
      <JobInfo Title="" Company="" Department=""/>
      <Phone Description="" Number=""/>
      <Email Description="" Address=""/>
      <Website Description="" URL=""/>
      <Note Note=""/>
   </Contact>
   <Contact/>
      ...
   <Contact/>
</Root>
```

## 18.8 Calendar Sample Files

In order to simulate the calendar of a device you can use a sample calendar file that has the XML structure shown in the listing below. You can then specify, in the Simulation tab of the Options dialog[1663], that this file is to be used for simulations involving the Access Calendar[673] action. Alternatively, you can use your Microsoft Outlook calendar for the simulation. The setting that specifies which of these two alternatives to use is in the Simulation pane of the Options dialog[1663].

⊟ *Template for sample calendar file*

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
   <Calendar Id="1" Name="Business">
      <Event Id="1" Title="Quarterly Meeting" Start="2018-04-04" End="2018-04-04"
AllDay="true()" Location="Meeting Room 2">
         <Attendee Name="Bob" Status="Accepted" Type="Required"
Relationship="Speaker"/>
      </Event>
      <Event Id="2" Title="New Customer Lunch" Start="2018-05-14T12:30:00" End="2018-
05-14T14:00:00" Location="Sushi Restaurant">
         <Attendee Name="Alice" Status="Accepted" Type="Optional"
Relationship="Attendee"/>
      </Event>
   </Calendar>
   <Calendar Id="2" Name="Private">
      <Event Id="1" Title="Family Dinner" Start="2018-05-18T19:00:00" End="2018-05-
18T23:00:00" Location="Home"/>
      <Event Id="2" Title="Summer Vacation" Start="2018-07-09" End="2018-07-22"
AllDay="true()" Location="Home"/>
   </Calendar>
</Root>
```

**Note:** In the Microsoft Outlook calendar, attendee status is visible only to the organizer of the event. So you will not be able to see the attendee status of events that were not organized by you.

# 18.9 Service Trigger Simulations

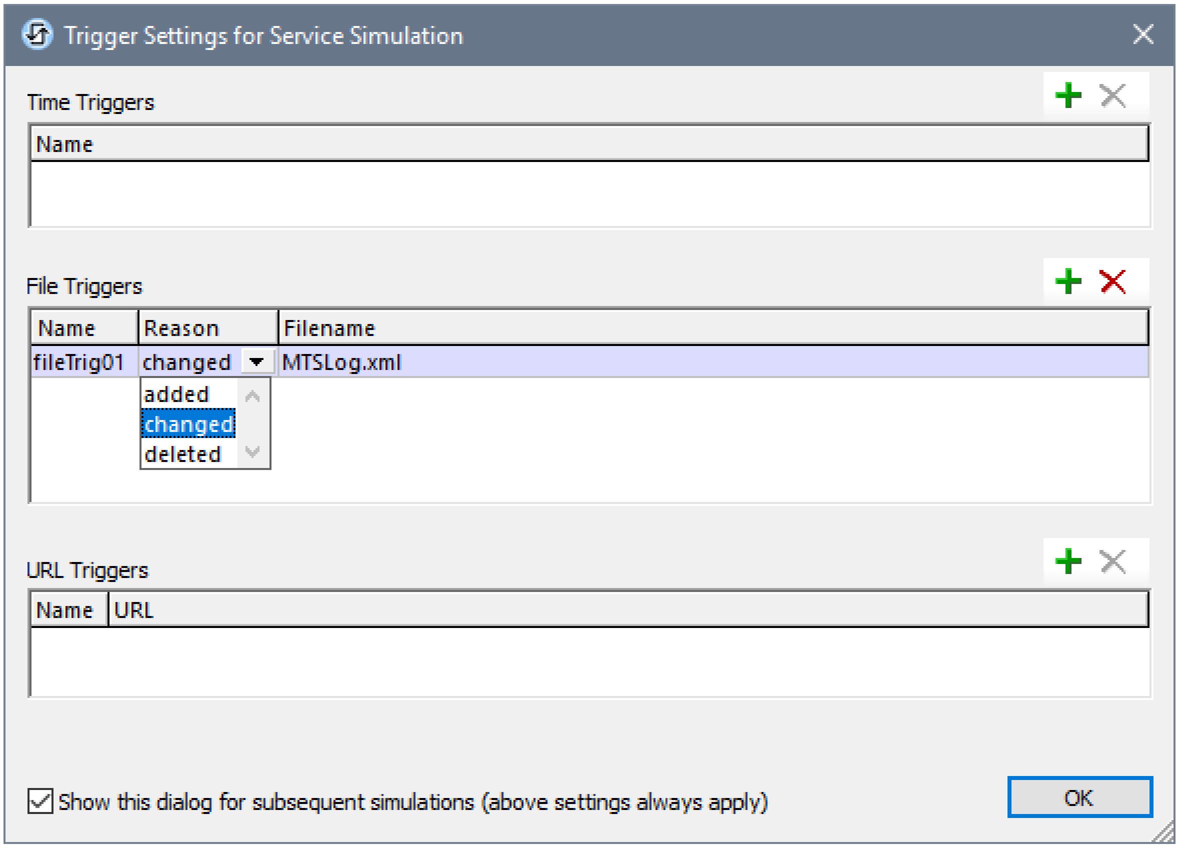
When a server service is simulated, the actions defined for the service are executed. If any of these actions make use of data in the `$MT_SERVICE`[1545] page source tree, then, for the simulation, you will need to manually create the data in the tree. This is because in a real-use scenario the data in the **`$MT_SERVICE`** page source is generated at run time from the service's trigger information—data that is not generated during a simulation.

⊟ *Structure of $MT_SERVICES page source*

```
<Root>
   <Triggers>
      <File name="" filename="" reason=""/>
      <URL name="" url=""/>
      <Timer name=""/>
   </Triggers>
</Root>
```

For simulations, you can manually create the data in the **`$MT_SERVICE`** tree as follows:

1. Start the simulation as usual: designer[1356] or server[1362]. The Trigger Settings for Service Simulation dialog appears.
2. This dialog contains a pane for each type of trigger. *(The pane for File System triggers is shown in the screenshot below.)* Add an entry for each individual trigger you want to simulate (by clicking the **Add** icon). (If you do not add any entry, the nodes of the **`$MT_SERVICE`** page source will be empty during a simulation. If no XPath expression in the definition of service actions accesses a node from the page source, then it is irrelevant whether the **`$MT_SERVICE`** tree contains any data or not.)
3. Enter the values with which you want to simulate the trigger's actions; each trigger is identified by its **`Name`** attribute. For example, notice, in the screenshot below, that you can enter three values for the File System trigger. These translate to the attribute values of a **`File`** element of the **`$MT_SERVICE`** tree (*see the structure of the `$MT_SERVICE` tree above*). If you want to use the server for a simulation[1362], make sure that the names you enter for triggers match the names of triggers on the server.



Trigger Settings for Service Simulation

Time Triggers

| Name |
|---|

File Triggers

| Name | Reason | Filename |
|---|---|---|
| fileTrig01 | changed | MTSLog.xml |

URL Triggers

| Name | URL |
|---|---|

☑ Show this dialog for subsequent simulations (above settings always apply)

4. If you wish to use these values for subsequent service simulations without having to see this dialog, then uncheck the *Show this dialog* option at the bottom of the dialog. (If at a later time you wish to show this dialog when you start a simulation, go to the *Simulations* tab of the Options dialog (**Tools | Options**[1663]), and check the option *Show dialog for triggers when simulating service*.
5. Click **OK**. The service will be simulated using the **`$MT_SERVICE`** page source values that you specified in the dialog (*described above*).

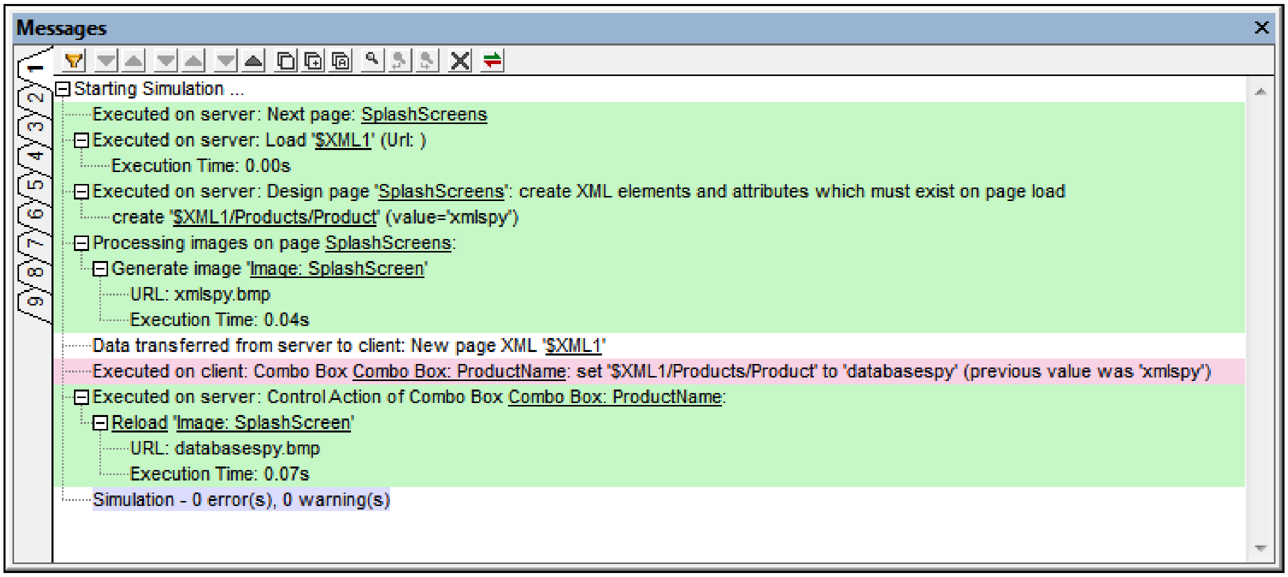
For a detailed description of how to create services, see the section Server Services[1543].

## 18.10 Messages Pane

While the simulation progresses, the Messages Pane 1385 of the MobileTogether Designer GUI provides a detailed and step-by-step report of all the solution-related activity taking place, and the time taken for each action. You can clearly see what is happening at each step in the progress of the workflow: what is executed on the server, what on the client, what is the order of the flow of actions, and why the actions happen the way they do. Server and client actions are displayed with different background colors. Errors are flagged, and warnings and traces are also displayed. All of this is absolutely necessary and extremely useful for testing and debugging design files.

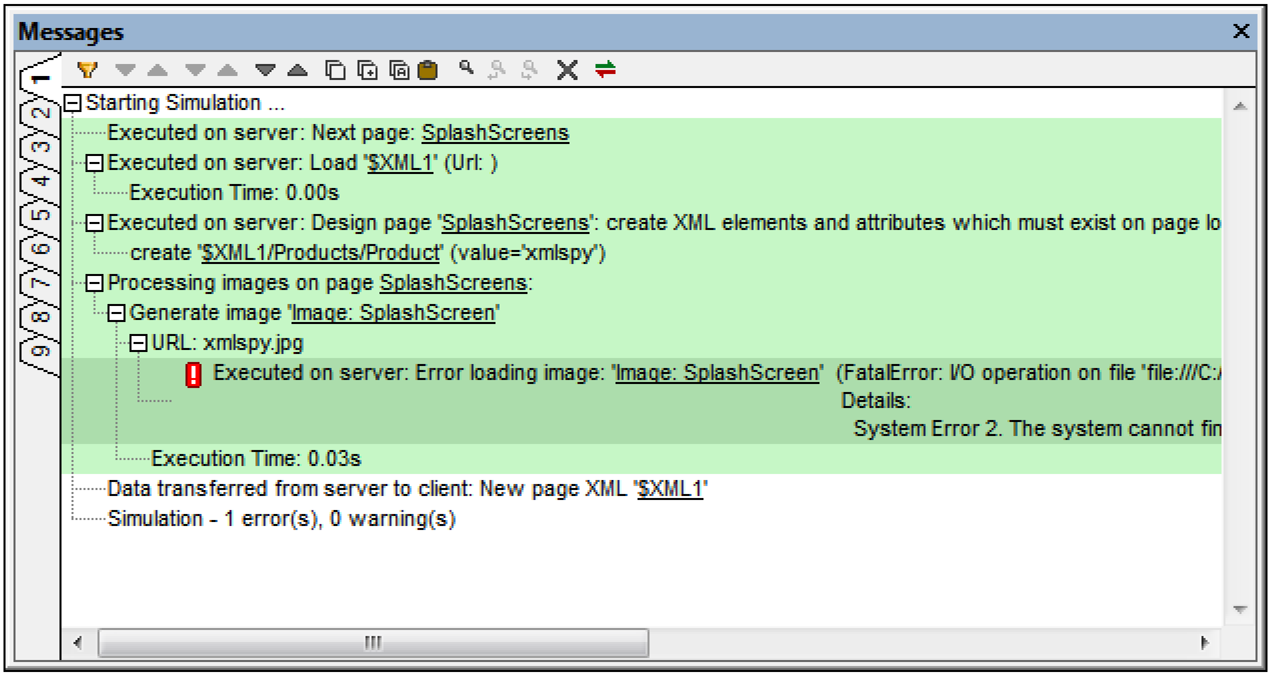
- Simulation on the designer reports activities on both server and client.
- Simulation on the server reports client messages.
- Simulation on the client reports server messages.

The screenshot below shows a simulation that was completed without any error. At each step of the workflow, messages detailing the actions taking place on server and/or client are displayed, as well as the time taken for the different executions. Messages about actions that are executed on the server and on the client are displayed in different background colors (green and pink, respectively, in the screenshot below). This helps you to quickly see where the various actions are taking place. (The background colors can be modified according to your preference 278.) Links in the Messages Pane 1385 take you to the relevant component in the design or the Page Sources Pane 270. Hovering over a message pops up additional information about that particular workflow activity. When the simulation ends, a summary of errors and warnings is displayed (*highlighted in the screenshot below*).



```
Starting Simulation ...
  Executed on server: Next page: SplashScreens
  Executed on server: Load '$XML1' (Url: )
    Execution Time: 0.00s
  Executed on server: Design page 'SplashScreens': create XML elements and attributes which must exist on page load
    create '$XML1/Products/Product' (value='xmlspy')
  Processing images on page SplashScreens:
    Generate image 'Image: SplashScreen'
      URL: xmlspy.bmp
      Execution Time: 0.04s
  Data transferred from server to client: New page XML '$XML1'
  Executed on client: Combo Box Combo Box: ProductName: set '$XML1/Products/Product' to 'databasespy' (previous value was 'xmlspy')
  Executed on server: Control Action of Combo Box Combo Box: ProductName:
    Reload 'Image: SplashScreen'
      URL: databasespy.bmp
      Execution Time: 0.07s
  Simulation - 0 error(s), 0 warning(s)
```

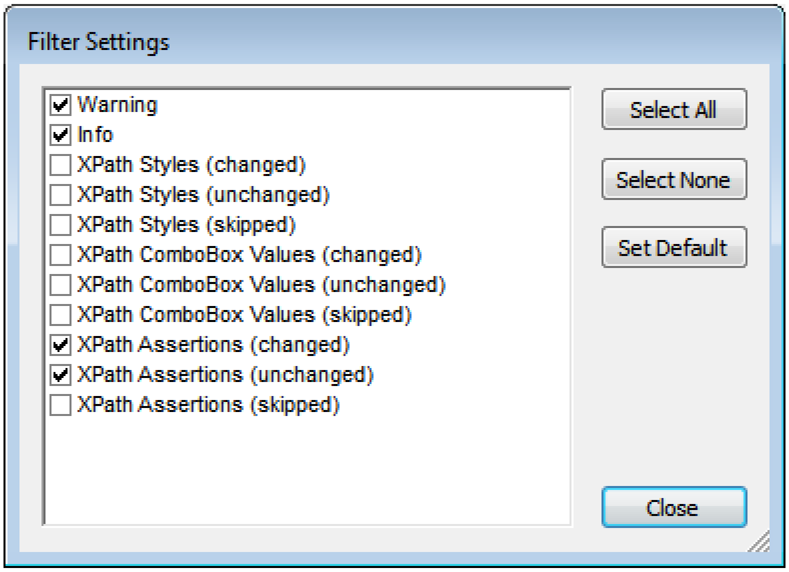
If an error occurs, it is flagged with a red icon (*see screenshot below*).



```
Starting Simulation ...
    Executed on server: Next page: SplashScreens
    Executed on server: Load '$XML1' (Url: )
        Execution Time: 0.00s
    Executed on server: Design page 'SplashScreens': create XML elements and attributes which must exist on page lo
        create '$XML1/Products/Product' (value='xmlspy')
    Processing images on page SplashScreens:
        Generate image 'Image: SplashScreen'
            URL: xmlspy.jpg
                Executed on server: Error loading image: 'Image: SplashScreen'  (FatalError: I/O operation on file 'file:///C:/
                                                                               Details:
                                                                                System Error 2. The system cannot fin
            Execution Time: 0.03s
    Data transferred from server to client: New page XML '$XML1'
    Simulation - 1 error(s), 0 warning(s)
```

*Filtering messages*

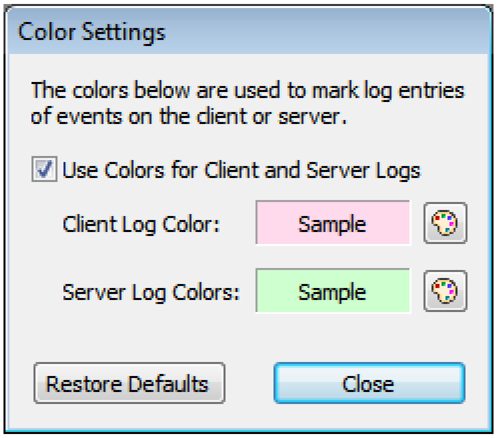
You can specify what kind of messages are displayed in the Messages Pane. To do this, click the **Filter** button in the toolbar of the Messages Pane (*screenshot above*). This displays the Filter Settings dialog (*screenshot below*). Select the message types you want to display and click **Close**. This feature can be very useful, for example, if there are too many messages and you wish to focus on just one type of message.



- ☑ Warning
- ☑ Info
- ☐ XPath Styles (changed)
- ☐ XPath Styles (unchanged)
- ☐ XPath Styles (skipped)
- ☐ XPath ComboBox Values (changed)
- ☐ XPath ComboBox Values (unchanged)
- ☐ XPath ComboBox Values (skipped)
- ☑ XPath Assertions (changed)
- ☑ XPath Assertions (unchanged)
- ☐ XPath Assertions (skipped)

*Color settings*

For messages that are displayed during simulations, different colors can be set for actions that take place on the server and on the client. If you set clearly distinguishable colors, you can very easily follow the workflow in the Messages Pane. This can be of great help in debugging. To set customs colors, click the **Colors** button in the toolbar of the Messages Pane (*screenshot above*). This displays the Color Settings dialog (*screenshot below*), in which you can set the colors you want.



Color Settings

The colors below are used to mark log entries of events on the client or server.

☑ Use Colors for Client and Server Logs

Client Log Color: Sample

Server Log Colors: Sample

For more information, see the Messages Pane[278].

# 19 MT Debugger

MT Debugger comprises two debuggers:

- Actions Debugger[1390], which is opened when an action that has been selected for debugging is encountered during a simulation.
- XPath Debugger[1398], which is built into the XPath/XQuery Window[1244]. You can access the XPath Debugger when editing XPath expressions in the design or when running simulations.

## Usage overview

MT Debugger is started from within a simulation. The broad usage steps are listed below:

1. Before running the simulation, set up debugging breakpoints on actions and/or XPath expressions if you want to debug these during the simulation
2. Start the simulation.
3. Select the Debugger mode[1389] you want to use: Breakpoints or Actions. In Breakpoints Mode, the simulation stops at all breakpoints (actions and XPath expressions) and opens the respective debugger. In Actions Mode, the simulation stops at the set of actions defined on the next event that is triggered in the Simulator. If no mode is selected, then the simulation proceeds without any debugging.

# 19.1 Debugger Modes

MT Debugger modes are set in the Simulator. There are three debugger modes:

- *Breakpoints Mode:* In this mode, the simulation will stop at breakpoints. If the breakpoint has been defined on an action, Actions Debugger[1390] will open and the action will be displayed in it. If the breakpoint is defined on an XPath expression, XPath Debugger[1398] will open and the expression will be displayed in it. You can then start debugging the action or expression.
- *Actions Mode:* In this mode, the simulation will stop at the set of actions defined on the next event that is triggered in the Simulator. The Actions Debugger[1390] will open, and the event's actions will be displayed in it.
- *Errors Mode:* In this mode, the simulation will stop when it encounters XPath errors.

Note the following points:

- Only one mode can be selected at a time.
- You can change modes during a simulation.
- After each simulation, the mode selection is removed. So the mode must be selected afresh for each simulation.

## Select the debugger mode

You can select a debugger mode either before or after starting a simulation:

- Before starting the simulation: Select the mode via the the MobileTogether Designer toolbar or the Debug[1626] menu.
- After starting the simulation: Select the mode in the Simulator toolbar[1355].

In both toolbars, the icons and commands are the same *(see below)*.

⊟ *MT Debugger modes*

| | | |
|---|---|---|
| | **Stop at Next Error** | Stops the simulation at XPath errors (Errors Mode) |
| | **Stop at Next Breakpoint** | Stops the simulation at breakpoints (Breakpoints Mode) |
| | **Stop at Next Action** | Stops the simulation at actions of the next event that is triggered (Actions Mode) |

# 19.2 Actions Debugger

The Actions Debugger enables you to debug the actions of a control event[665] or page event[389]. It is opened when an action that has been selected for debugging is encountered during a simulation. The debugging of actions enables you to do the following: (i) view the callstack of actions, (ii) view how the values of variables are updated during the execution of actions, and (iii) set watch expressions to test or investigate aspects of the action execution.

## Usage

Usage of the Actions Debugger consists of two steps:

1. Select the action/s to be debugged[1390]
2. Start a simulation and run the Actions Debugger[1392] on the selected actions

## Select the action/s to debug

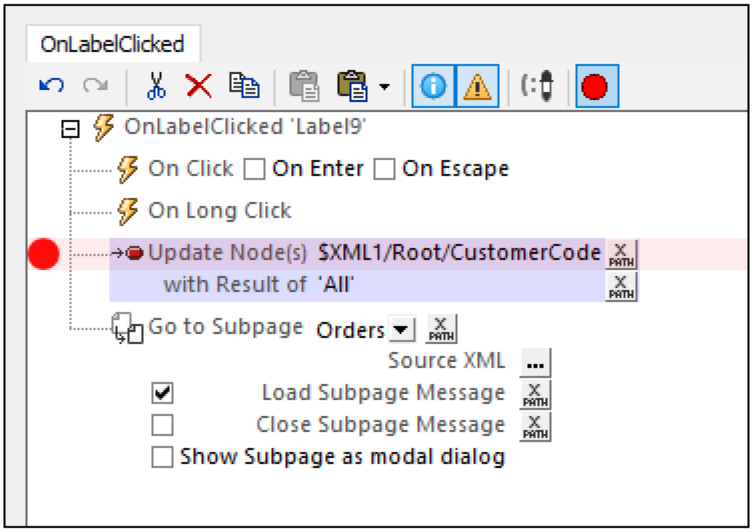
Selection is done in two ways:

- By setting a breakpoint on each action you want to debug. If you then run a simulation in Breakpoints Mode, the Actions Debugger will automatically open to debug actions that have breakpoints. (In Breakpoints Mode, XPath Debugger[1398] will be opened to debug any XPath expression that contains a breakpoint.)
- By directly selecting, during a simulation, the next action to debug. This is Actions Mode.

*Setting breakpoints on actions*

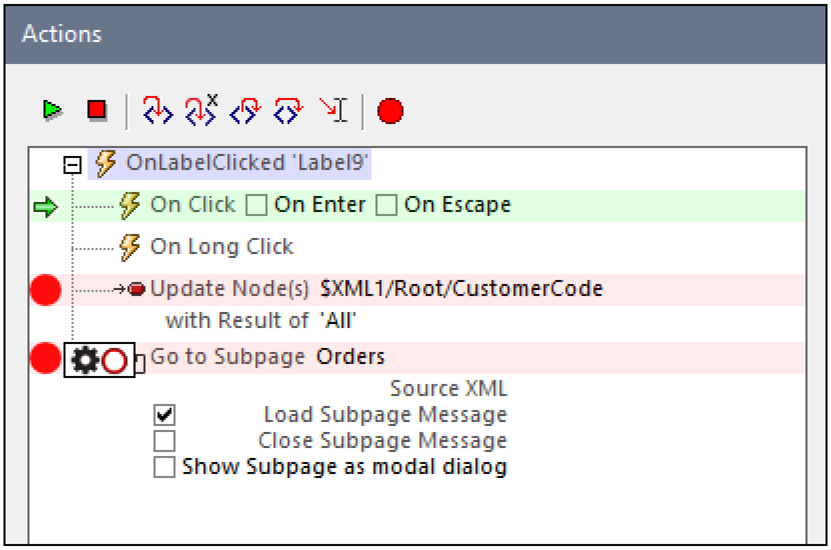
You can set a breakpoint on an action in two ways:

- On an action's definition[667], by clicking the toolbar command **Toggle Breakpoint** *(screenshot below)* to toggle the breakpoint on. To remove the breakpoint, click **Toggle Breakpoint** again.



OnLabelClicked 'Label9'
On Click □ On Enter □ On Escape
On Long Click
● Update Node(s) $XML1/Root/CustomerCode
with Result of 'All'
Go to Subpage Orders
Source XML
☑ Load Subpage Message
□ Close Subpage Message
□ Show Subpage as modal dialog

- During an action debugging session you might find that you want to set or remove breakpoints for future debugging sessions. This is possible without you having to close your current debugging session. In the Actions Debugger *(screenshot below),* select the action for which you want to set/remove a breakpoint, and click the toolbar command **Toggle Breakpoint**.



OnLabelClicked 'Label9'
On Click □ On Enter □ On Escape
On Long Click
Update Node(s) $XML1/Root/CustomerCode
with Result of 'All'
Go to Subpage Orders
Source XML
☑ Load Subpage Message
□ Close Subpage Message
□ Show Subpage as modal dialog

The following options are available when you place the mouse cursor over a breakpoint's symbol (the red circle):

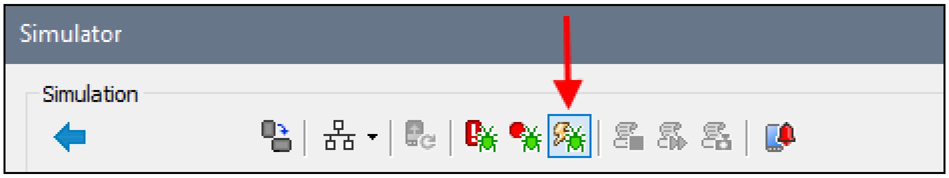
- Click the gear symbol in the popup to make the breakpoint conditional. The condition is specified as follows: (i) via an XPath expression, and (ii) via the hit count (the number of times the action has been executed). For example, you can specify that the breakpoint is active when the the hit count is equal to 3, which will result in the breakpoint being active the third time the action is executed.
- Click the outlined-circle toggle in the popup to disable/enable the breakpoint.

A plus symbol appears inside the red circle symbol of the breakpoint when the breakpoint has been made conditional. This symbol disappears when the conditionality is removed.

**Note:** You can remove a breakpoint by clicking it, or by selecting the respective action and clicking the **Toggle Breakpoint** icon in the toolbar of the pane.

*Select an action during simulation*

Besides using breakpoints to select an action for debugging, you can also, during a simulation[1355], use the simulator's **Stop at Next Action** toolbar icon *(see screenshot below)* to select actions for debugging. The selected actions in this case will be the actions of the next event to be triggered in the simulation.



When the next event is triggered in the simulator, the Actions Debugger will open, and you can start debugging the actions that were defined for that event. How the Actions Debugger works[1392] is explained below.

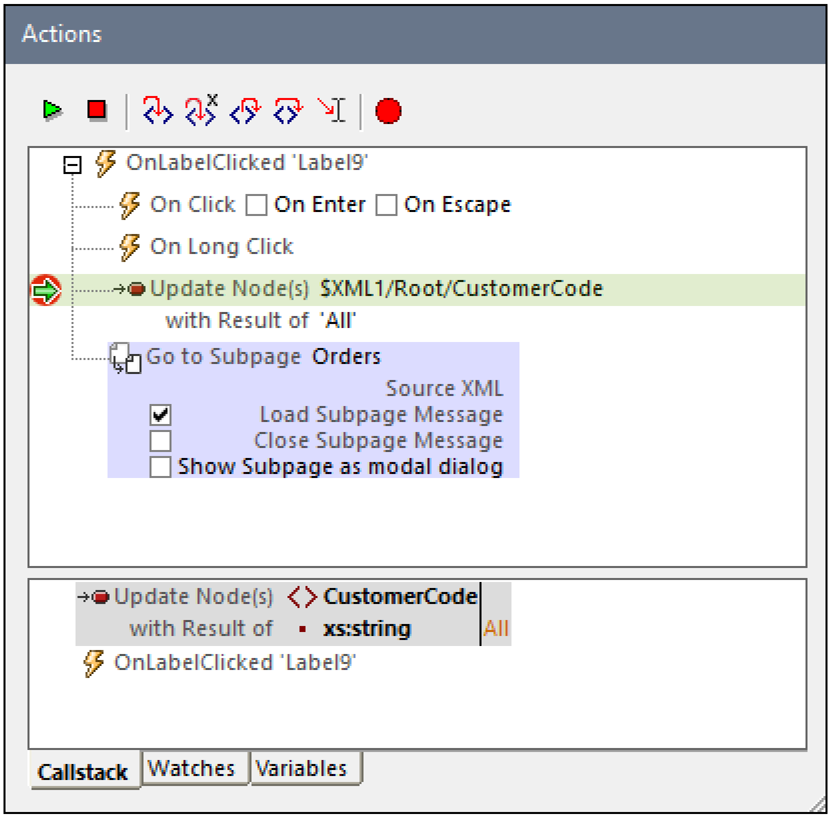
## Running the Actions Debugger

The Actions Debugger can run in Breakpoints Mode (debugs actions having breakpoints) or Actions Mode (debugs actions of the next event to be triggered in the simulation).

After you have set any breakpoints[1390] that you want to set on actions *(see section above)*, in the simulator select the debugger mode you want (Breakpoints or Actions)[1389] and then start or resume the simulation. When the simulation reaches an action selected for debugging[1390], the Actions Debugger appears.

The Actions Debugger *(screenshot below)* consists of two panes:

- An upper pane, which displays the action/s that have been selected for debugging. The screenshot below, for example, shows the two actions that have been set on the `OnLabelClicked` event of a Label control named `Label9`. These actions are an *Update Node* action and a *Go to Subpage* action.
- A lower pane, which has three tabs: (i) *Callstack*, (ii) *Watches*, (iii) *Variables* (described below[1393]).



- You can start debugging by using the toolbar icons
- The action being currently debugged is indicated by a green highlight and green arrow at left *(see screenshots in this topic)*
- The toolbar icons *(see screenshot above)* are, from left to right:
  - ❖ *Resume Debugging / Go:* Starts/Continues the debugging.
  - ❖ *Stop Debugging:* Stops debugging and closes the Actions Debugger. You will be prompted to choose whether you want to (i) stop the debugging and the simulation; (ii) stop the debugging and the simulation, and edit the action; (iii) cancel the stop action and continue with debugging.
  - ❖ *Step Into Action (F11):* Proceeds through the action execution, one step at a time.

- *Step Into XPath (Ctrl+Shift+F11):* Opens the XPath Debugger 1398 and displays the XPath expression of the action.
- *Step Out (Shift+F11):* Steps out of the current execution step, and goes to the parent step.
- *Step Over (Ctrl+F11):* Steps over descendant steps.
- *Run to Cursor:* Stops at the action currently selected by the cursor. (The currently selected action is highlighted in light blue; *see screenshot above*.) If a breakpoint occurs earlier, processing stops at the breakpoint/s. If the current selection cannot be reached, then processing proceeds to the end stopping only at breakpoints.
- *Toggle Breakpoint (F9):* Toggles on/off a breakpoint on the selected action. If MT Debugger is run in Breakpoints Mode 1389, then the simulation will be stopped at the selected action to debug.

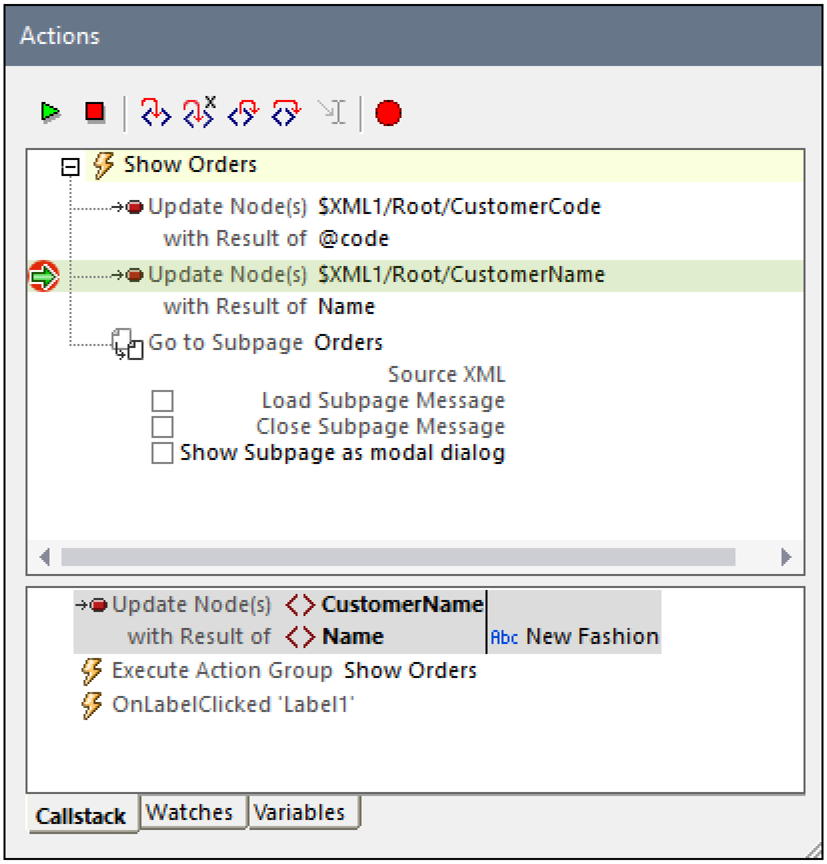
## The Callstack, Watches, and Variables panes

The three tabs of the lower pane display the results of the debugging. They are described below. At each step of debugging, the results in each tab are updated. You can switch between tabs to check the various results.

The three screenshots below show the Actions Debugger during a simulation of the SubPages-And-Visibility 188 tutorial. The debugger was in Actions Mode 1389 and the first customer, *New Fashion*, was clicked. This opened the Actions Debugger with the action of the `OnLabelClicked` control displayed (the label containing the *New Fashion* text). The action was an action group named *Show Orders*. On clicking **Step Into** in the Actions Debugger, we stepped into the *Show Orders* action group (the actions of which are shown in the screenshots below), and stepped past the first *Update Node* action. The screenshots were taken on stepping into the second *Update Node* action. You can open the SubPages-And-Visibility 188 tutorial, and try this out for yourself.

*Callstack pane*

Displays the actions that are executed and the results of the execution *(see screenshot below)*. The callstack displays the current action, in which the node **`$XML1/Root/CustomerName`** has been updated with the text content of the **`Name`** element.



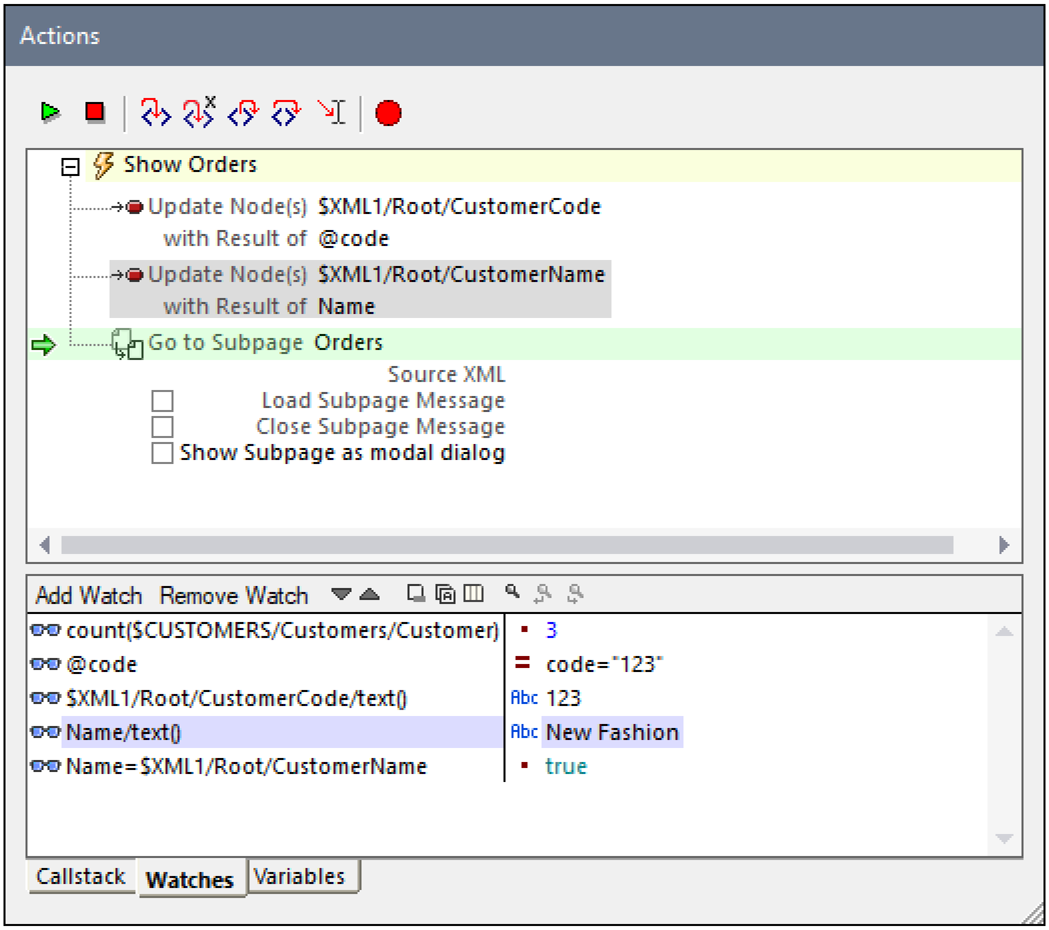
Actions

⊟ Show Orders
Update Node(s) $XML1/Root/CustomerCode
with Result of @code
Update Node(s) $XML1/Root/CustomerName
with Result of Name
Go to Subpage Orders
Source XML
□ Load Subpage Message
□ Close Subpage Message
□ Show Subpage as modal dialog

Update Node(s) <> CustomerName
with Result of <> Name | Abc New Fashion
Execute Action Group Show Orders
OnLabelClicked 'Label1'

*Watches pane*

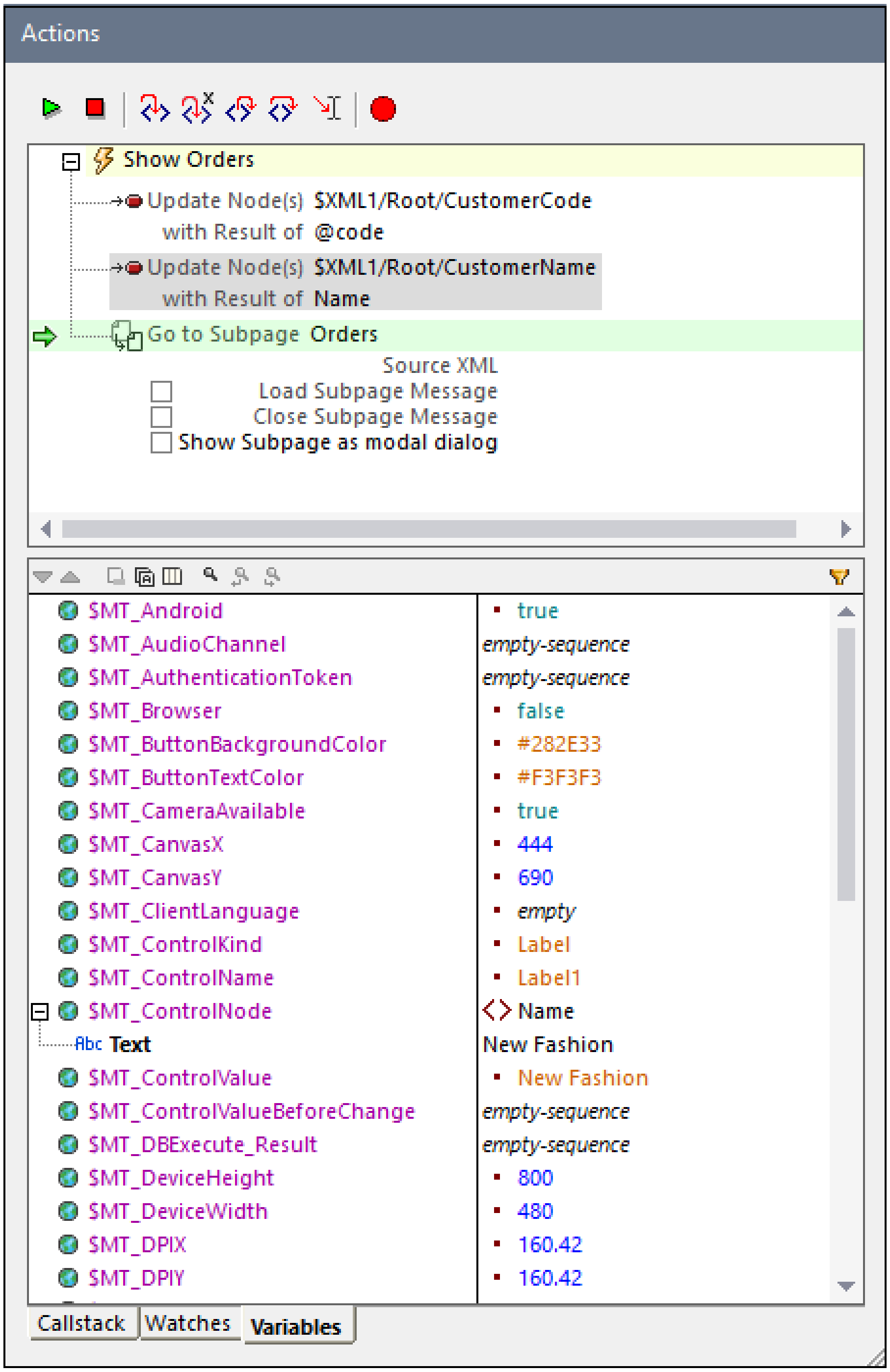
In the Watches pane, you can enter XPath expressions to display data, generate new data (such as, in the example below, the number of customers), and test whether certain conditions are true. Click **Add Watch** to add a new watch expression, enter the XPath expression, and press **Return**.



Show Orders
Update Node(s) $XML1/Root/CustomerCode
with Result of @code
Update Node(s) $XML1/Root/CustomerName
with Result of Name
Go to Subpage Orders
Source XML
□ Load Subpage Message
□ Close Subpage Message
□ Show Subpage as modal dialog

| | |
|---|---|
| count($CUSTOMERS/Customers/Customer) | 3 |
| @code | code="123" |
| $XML1/Root/CustomerCode/text() | 123 |
| Name/text() | New Fashion |
| Name=$XML1/Root/CustomerName | true |

*Variables pane*
The Variables pane display all the design variables that are currently in scope, including Static Global Variables 1300 and Dynamic Local Variables 1304. This information can be very useful for building and debugging expressions defined inside actions.



Actions

Show Orders
Update Node(s) $XML1/Root/CustomerCode
with Result of @code
Update Node(s) $XML1/Root/CustomerName
with Result of Name
Go to Subpage Orders
Source XML
□ Load Subpage Message
□ Close Subpage Message
□ Show Subpage as modal dialog

| | |
|---|---|
| $MT_Android | true |
| $MT_AudioChannel | *empty-sequence* |
| $MT_AuthenticationToken | *empty-sequence* |
| $MT_Browser | false |
| $MT_ButtonBackgroundColor | #282E33 |
| $MT_ButtonTextColor | #F3F3F3 |
| $MT_CameraAvailable | true |
| $MT_CanvasX | 444 |
| $MT_CanvasY | 690 |
| $MT_ClientLanguage | *empty* |
| $MT_ControlKind | Label |
| $MT_ControlName | Label1 |
| $MT_ControlNode | Name |
| Text | New Fashion |
| $MT_ControlValue | New Fashion |
| $MT_ControlValueBeforeChange | *empty-sequence* |
| $MT_DBExecute_Result | *empty-sequence* |
| $MT_DeviceHeight | 800 |
| $MT_DeviceWidth | 480 |
| $MT_DPIX | 160.42 |
| $MT_DPIY | 160.42 |

## Closing the Actions Debugger

You can close the Actions Debugger in the following ways:

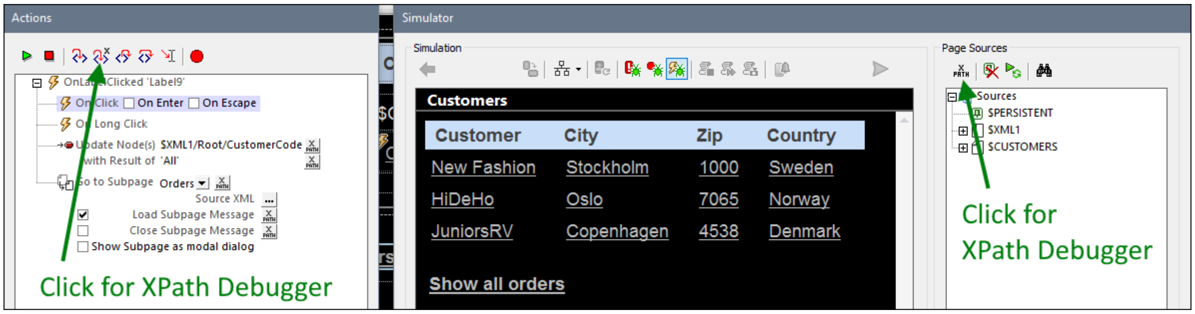
- Click **Stop Debugging**.

- Click through the action/s using the **Step Into**, **Step Out Of**, and **Step Over** toolbar buttons till the action/s of the current event have all been executed in the debugger.
- In the Evaluator, toggle off the Actions Mode 1389.

# 19.3 XPath Debugger

You can access XPath Debugger in the following contexts:

1. When editing an XPath expression in the design. For example, when entering expressions to set values of styles and properties 274, you can debug expressions by testing them against an XML file that you load into the Debugger 1252.
2. When running a simulation 1355: Enter a new expression at any time during the simulation to debug it against an XML file that you load into the Debugger 1252. To start the XPath Debugger, click the **Evaluate XPath** button in the Page Sources pane *(see screenshot below)*.
3. When running a simulation 1355: In the Actions Debugger 1390, click the **Step Into XPath** button to debug the current action's XPath expression against the action's page sources *(see screenshot below)*.



Customers

| Customer | City | Zip | Country |
|---|---|---|---|
| New Fashion | Stockholm | 1000 | Sweden |
| HiDeHo | Oslo | 7065 | Norway |
| JuniorsRV | Copenhagen | 4538 | Denmark |

Show all orders

How to use XPath Debugger 1252 is explained in the XPath/XQuery Window 1244 section.

## Setting breakpoints on XPath expressions

You can set breakpoints on XPath expressions in XPath Debugger 1252 after opening XPath Debugger via any of the three contexts listed above. Furthermore, if breakpoints have been set on expressions 1257, you can debug these expressions directly from a simulation by starting MT Debugger 1388 in Breakpoints Mode 1389. In this mode the simulation will stop at all breakpoints, whether they have been set on actions or XPath expressions, opening the appropriate debugger: Actions Debugger 1390 or XPath Debugger 1252.

# 20 Automated Testing

The Automated Testing feature enables you to compare two test runs (which are essentially simulations[1355]) to detect differences in the design, page source data, component styles or layout, or solution environment.

The process works as follows: First, a base test run (or **test case**) of a design is recorded. This test case progresses through certain user and design actions[667]. The test case is subsequently played back with different parameters (for example, with different page source data, or on a different version of a device-OS). If the playback in MobileTogether Designer returns differences from the test case, then the playback is recorded. A recorded playback is called a **test run**—as opposed to the original test *case*. The test run can then be compared with its originating test case. If any issues are identified, these can be addressed. Furthermore, a test case for a design can be deployed, together with the design, to MobileTogether Server. This enables test cases to be downloaded to multiple client devices for playback. Playbacks on client devices are saved to the server, and can be retrieved in MobileTogether Designer for comparison.

The typical Automated Testing scenario would progress as follows:

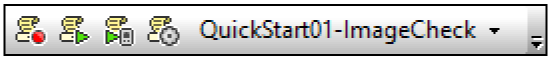
1. *Record a test case.* The recorded test case can be played back in a different environment.
2. *Playback a test case.* Playbacks are saved as test runs of its test case. If the playback is in MobileTogether Designer, then only those playbacks that return a difference are stored as test runs. If a test case is deployed to MobileTogether Server and played back on a client device, then all these client playbacks are stored on the server.
3. *The test run is compared with its originating test case.* Comparisons are carried out in MobileTogether Designer. The differencing level can be configured, and the differences can be examined in detail[1414]. Test runs that are returned from client playbacks (and stored on the server) will have to be retrieved to MobileTogether Designer for comparisons.

## Using Automated Testing for quickly carrying out routine steps

In cases where certain routine steps need to be carried out every time you run a simulation, these steps can be recorded as a test case and subsequently played back. For example, the design might prompt the user to enter log-in details or other items of data that do not change. If the entry of this data takes up much time, the data-entry steps can be recorded in a test case. Subsequently, you can play back the test case to quickly complete these routine steps and then carry out additional test-steps manually. Used in this way, Automated Testing can help you to save time during the design phase.

## Automated Testing menu commands

The commands to run the Automated Testing feature are in the **Run**[1617] menu. They are also available via the Automated Testing toolbar (*screenshot below*).



| | |
|---|---|
| | *Record New Test Case:* Starts a new test case in the Simulator[1355] and records user actions. When recording stops, you are prompted to give the recording a name and save it as a test case. *See Recording a Test Case[1401] for details*. |
| | *Playback Test Case:* Plays back the test case that is selected in the *Available Test Cases for Playback* combo box. If the playback returns differences from the test case, then the playback is saved. *See Playing Back a Test Case[1403]*. |

| | |
|---|---|
| | *Trial Run Test Cases on Client:* Plays back, on a connected client, the test case that is selected in the *Available Test Cases for Playback* combo box. If the playback returns differences, then the playback is saved. *See Playing Back a Test Case*[1403]. |
| | *Manage Test Cases and Runs:* Displays the *Manage Test Cases and Runs*[1401] dialog. |

The *Available Test Cases for Playback* combo box is is displayed only after a test case has been recorded. It displays all the recorded test cases. Select the test case you want to play back. The test case that is selected here will be run when **Playback Test Case** or **Trial Run Test Cases on Client** is clicked.

## In this section

This section is organized as follows:

- Recording a Test Case[1401]
- Playing Back a Test Case[1403]
- Managing Test Cases and Runs[1406]
- Deploying Test Cases to Server[1410]
- Comparing Test Runs[1414]

# 20.1 Recording a Test Case

This action creates a test case, which can be used as the base case for comparisons.

| | |
|---|---|
| | *Record New Test Case:* Starts a new test case in the Simulator[1355] and records user actions. When recording stops, you are prompted to give the recording a name and save it as a test case. |
| | *Stop Recording Test Case:* Stops recording, and opens the Recorded Test Case Confirmation dialog, in which you specify the name of the recorded test case. |

Record a test case as follows:

1. Make the design that you want to test the active design.
2. Click the toolbar icon **Record New Test Case** (*see icon above*).
3. In the Simulator[1355] window that appears, carry out the steps you want to include in the test. You can also take snapshots manually if this option has been enabled in the recording options (*see below*).
4. When you have completed the test case, click **Close** (*at bottom right*) or the toolbar icon **Stop Recording Test Case** (*see icon above*).
5. In the Recorded Test Case Confirmation dialog that appears, enter the name of the test case, and click **Save**.

For each design, you can record as many test cases as you like.

## Recording options

Options for recording and playback are available at the bottom of the Manage Test Cases and Runs dialog[1406]. Since playbacks that show differences are automatically recorded and stored, the recording options you set here are used for playbacks as well.

*Recording options*
The following recording options are available:

- *Logging of design actions:* Design actions[667] are actions that are not explicitly triggered by the user. (An example would be the automatic saving of data to a page source.) If this option is checked (the default), then design actions are logged.
- *Automatically taking snapshots after each step:* Check this option to enable automatic snapshots after each user action. Snapshots will be taken of the snapshot-items selected by you in the inclusion options (*see next point*).
- *What to include in snapshots:* These options apply to the *recording of test cases*—not to the playback of test runs. Select what you want to include in snapshots. The options are: page sources, styles, and client views (the layout coordinates of design components on clients). If you select at least one of these snapshot options and de-select *Make Snapshot Automatically*, then the **Record Snapshot** button in the Simulator[1355] will be enabled during the recording of test cases, and you can take snapshots at any time that you want.

*Playback options*
The following playback options are available:

- *Test case speed:* You can select at what speed to play back a test case. If you select *Step-by-Step*, then the **Playback Next Step** button in the Simulator[1355] will be enabled during playback, and you can run through the test at your own pace; click **Playback Next Step** to progress one step at a time.

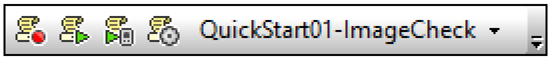
- *Reset persistent data:* Each test case has a *Reset* check box for resetting persistent data to original values. If checked, then, when a test case is played back, persistent data on the client is reset to their original values.

# 20.2 Playing Back a Test Case

After a test case has been recorded (*see Recording a Test Case*[1401]), you can play back that test case with different parameters (for example, with different page source data, or on different versions of a device-OS). If the playback returns differences, then the playback is automatically saved in the design when it is closed; otherwise it is not saved. A saved playback is displayed in the Manage Test Cases and Runs dialog[1406] as a **test run** that is based on its originating **test case** (notice the terminology; *see Managing Test Cases and Runs*[1406] for more information).

## Playback in designer and client

You can play back a test case in MobileTogether Designer or on a client device. To do this, click the corresponding icon in the toolbar (*see screenshot below*). These are, respectively, **Playback Test Case** and **Trial Run Test Cases on Client**.



| | |
|---|---|
| | *Playback Test Case:* Plays back the test case that is selected in the *Available Test Cases for Playback* combo box. If the playback returns differences from the test case, then the playback is saved. |
| | *Trial Run Test Cases on Client:* Plays back, on a connected client, the test case that is selected in the *Available Test Cases for Playback* combo box. The playback is saved (whether there are differences or not). |

## Playback options

Options for recording and playback are available at the bottom of the Manage Test Cases and Runs dialog[1406]. Since playbacks that show differences are automatically recorded and stored, the recording options you set here are used for playbacks as well.

*Recording options*
The following recording options are available:

- *Logging of design actions:* Design actions[667] are actions that are not explicitly triggered by the user. (An example would be the automatic saving of data to a page source.) If this option is checked (the default), then design actions are logged.
- *Automatically taking snapshots after each step:* Check this option to enable automatic snapshots after each user action. Snapshots will be taken of the snapshot-items selected by you in the inclusion options (*see next point*).
- *What to include in snapshots:* These options apply to the *recording of test cases*—not to the playback of test runs. Select what you want to include in snapshots. The options are: page sources, styles, and client views (the layout coordinates of design components on clients). If you select at least one of these snapshot options and de-select *Make Snapshot Automatically*, then the **Record Snapshot** button in the Simulator[1355] will be enabled during the recording of test cases, and you can take snapshots at any time that you want.

*Playback options*
The following playback options are available:

- *Test case speed:* You can select at what speed to play back a test case. If you select *Step-by-Step*, then the **Playback Next Step** button in the Simulator 1355 will be enabled during playback, and you can run through the test at your own pace; click **Playback Next Step** to progress one step at a time.
- *Reset persistent data:* Each test case has a *Reset* check box for resetting persistent data to original values. If checked, then, when a test case is played back, persistent data on the client is reset to their original values.

## Playing back a test case in MobileTogether Designer

To play back a test case in MobileTogether Designer, do the following:

1. In the *Available Test Cases for Playback* combo box (*see toolbar screenshot above*), select the test case that you want to play back. Note that you can play back test cases only, not test runs.
2. Click **Playback Test Case** (*see toolbar screenshot above*).

Playback will run in the Simulator 1355. If the playback returns differences, then the playback is automatically saved as a test run when it is closed.

**Note:** Message boxes are not displayed during playback in MobileTogether Designer.

## Playing back a test case on a client device

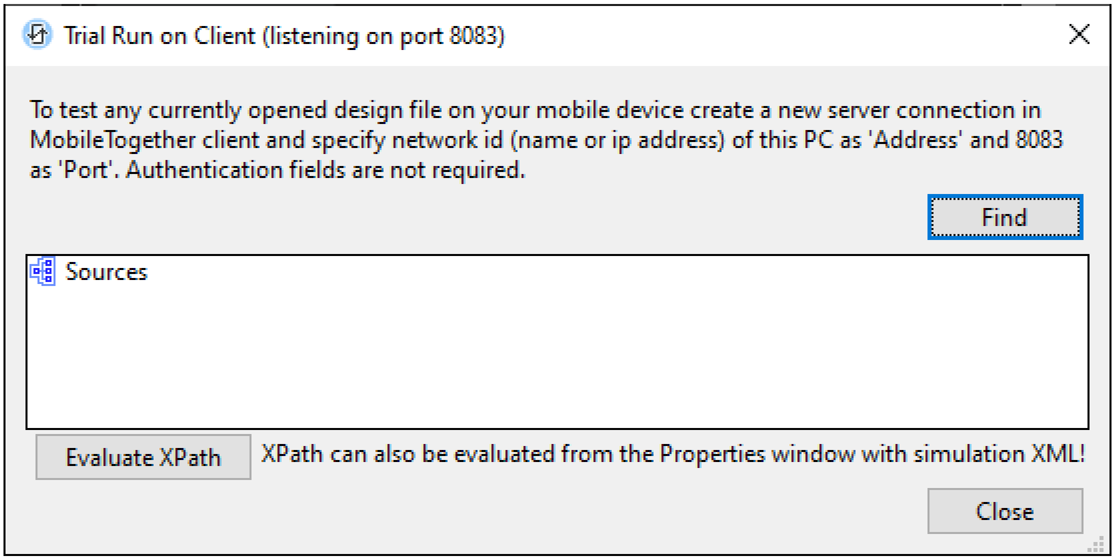
Playing back a test case on a client enables you to preview and check whether a given test case functions correctly on specific client devices. In this scenario, the MobileTogether Designer machine takes on the role of a MobileTogether Server. This means that you must set a separate port via which the Designer machine (acting as a server) will connect with the client.

The setup consists of two parts:

- In MobileTogether Designer, go to **Tools | Options | Trial Run on Client** 1663, and set up a separate port for communication with the client device: for example, **`8083`**.
- On your client device, add the Designer machine as a new server. You will need: (i) the machine's IP address (which you can find out with the DOS command **`ipconfig`**), and (ii) the port number you configured in the previous step.

To play back a test case, do the following:

1. In the *Available Test Cases for Playback* combo box (*see toolbar screenshot above*), select the test case that you want to play back. Note that you can play back test cases only, not test runs.
2. Click **Trial Run Test Cases on Client** (*see toolbar screenshot above*). The Trial Run on Client dialog opens (*screenshot below*).



Trial Run on Client (listening on port 8083)

To test any currently opened design file on your mobile device create a new server connection in MobileTogether client and specify network id (name or ip address) of this PC as 'Address' and 8083 as 'Port'. Authentication fields are not required.

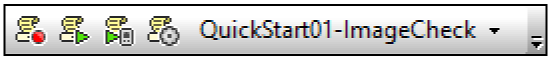
XPath can also be evaluated from the Properties window with simulation XML!

3. On the client device, connect to the Designer machine and refresh the solutions. On the client, you should see all the solutions that are currently active in MobileTogether Designer.
4. On the client device, start the solution for which you wish to play back a test case on the client.

The test case will be played back on the client. In MobileTogether Designer, the Trial Run on Client dialog (*screenshot above*) will be updated with page source data as the playback progresses. You can search in the page source data by clicking the **Find** button and entering the search string. The playback is automatically saved (whether there are differences or not) as a test run when it is closed. To check whether there were differences, click **Manage Test Cases and Runs**[1406] in the *Automated Testing* toolbar.

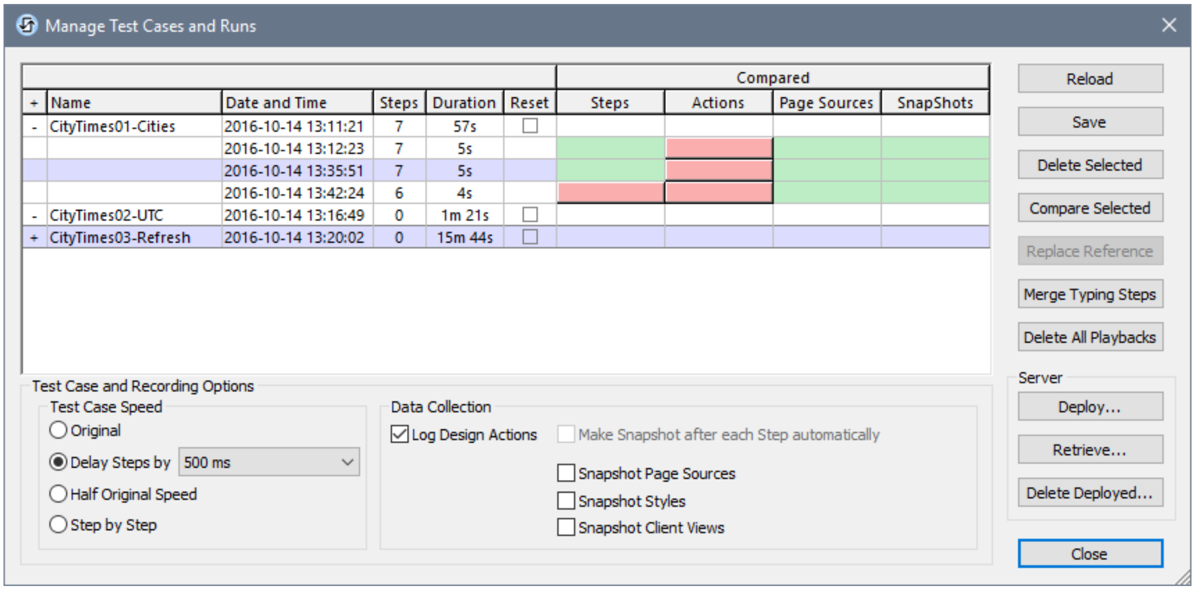
# 20.3 Managing Test Cases and Runs

You can open the Manage Test Cases and Runs dialog by clicking the **Manage Test Cases and Runs** icon in the Automated Testing toolbar (*see screenshot below*).



In the Manage Test Cases and Runs dialog (*screenshot below*) you can do the following:

- Set up recording options for test cases.
- Set up recording and playback options for subsequent test runs.
- Specify that the test runs of individual test cases start with their persistent data reset to original values.
- Reload and save MobileTogether Recording files (`.mtrecord` files). At any given time, there is at most one `.mtrecord` file per design file, and it has the same name as the design file (but a different file extension).
- Delete and compare test runs.
- Substitute a test case with one of its test runs. The test run takes on the role of the test case. Other test runs are deleted, and the selected test run becomes the new test case of that (now empty) group.
- Deploy a test case to MobileTogether Server, retrieve test runs from the server, and delete a test case or test run from the server.



| | | | | | | Compared | | | |
|---|---|---|---|---|---|---|---|---|---|
| + | Name | Date and Time | Steps | Duration | Reset | Steps | Actions | Page Sources | SnapShots |
| - | CityTimes01-Cities | 2016-10-14 13:11:21 | 7 | 57s | □ | | | | |
| | | 2016-10-14 13:12:23 | 7 | 5s | | 🟩 | 🟥 | 🟩 | 🟩 |
| | | 2016-10-14 13:35:51 | 7 | 5s | | 🟩 | 🟥 | 🟩 | 🟩 |
| | | 2016-10-14 13:42:24 | 6 | 4s | | 🟥 | 🟥 | 🟩 | 🟩 |
| - | CityTimes02-UTC | 2016-10-14 13:16:49 | 0 | 1m 21s | □ | | | | |
| + | CityTimes03-Refresh | 2016-10-14 13:20:02 | 0 | 15m 44s | □ | | | | |

Test Case and Recording Options

Test Case Speed
○ Original
● Delay Steps by 500 ms
○ Half Original Speed
○ Step by Step

Data Collection
☑ Log Design Actions
□ Make Snapshot after each Step automatically
□ Snapshot Page Sources
□ Snapshot Styles
□ Snapshot Client Views

The dialog displays each recorded test case and its associated test runs[1403] (playbacks[1403]), together with relevant data. The date and time is taken from the client device as are other relevant client settings, such as the language.

## Recording and playback options

Options for recording and playback are available at the bottom of the Manage Test Cases and Runs dialog[1406]. Since playbacks that show differences are automatically recorded and stored, the recording options you set here are used for playbacks as well.

*Recording options*
The following recording options are available:

- *Logging of design actions:* Design actions[667] are actions that are not explicitly triggered by the user. (An example would be the automatic saving of data to a page source.) If this option is checked (the default), then design actions are logged.
- *Automatically taking snapshots after each step:* Check this option to enable automatic snapshots after each user action. Snapshots will be taken of the snapshot-items selected by you in the inclusion options (*see next point*).
- *What to include in snapshots:* These options apply to the *recording of test cases*—not to the playback of test runs. Select what you want to include in snapshots. The options are: page sources, styles, and client views (the layout coordinates of design components on clients). If you select at least one of these snapshot options and de-select *Make Snapshot Automatically*, then the **Record Snapshot** button in the Simulator[1355] will be enabled during the recording of test cases, and you can take snapshots at any time that you want.

*Playback options*
The following playback options are available:

- *Test case speed:* You can select at what speed to play back a test case. If you select *Step-by-Step*, then the **Playback Next Step** button in the Simulator[1355] will be enabled during playback, and you can run through the test at your own pace; click **Playback Next Step** to progress one step at a time.
- *Reset persistent data:* Each test case has a *Reset* check box for resetting persistent data to original values. If checked, then, when a test case is played back, persistent data on the client is reset to their original values.

## Test cases and their associated test runs

The names of test cases are defined at the end of the recording process[1401]. In the Manage Test Cases and Runs dialog (*screenshot above*), you can change these names by double-clicking the name and editing it. Each test case and test run is identified by an internal ID. So if the test case/run has been saved to file (*see below*), a name change is recognized (on the basis of the internal IDs).

Playbacks that return a difference from the test case are stored as test runs that are associated with their base test case (the test case that was played back). Test runs are each automatically given a default name of `Test Run` when they are stored (*see screenshot above*). These names can be changed subsequently, by double-clicking the name and editing it.

In the Manage Test Cases and Runs dialog (*screenshot above*), test runs are considered to depend on their originating test case. So you can collapse and expand a test case to hide and show its associated test runs. If you delete a test case, then all its associated test runs are also deleted.

## Merge typing steps

This button is enabled when a test case—not a test run—is selected, and it applies to edit fields only. If an action exists for the **`OnTyping`** event of an edit field, then every keystroke in the edit field will have been recorded as a separate step. If you wish to merge all the keystrokes into a single step for the whole edit field, click **Merge Typing Steps**. Note that this merge cannot be undone.

## Resetting persistent data

Select the *Reset* check box of a test case (*see the Manage Test Cases and Runs dialog screenshot above*) to reset the design's persistent data to original values when this test case is played back.

## Saving and reloading test cases and runs

All test cases and their test runs can be saved in a file called ***`<mtdesignfilename>`*****`.mtrecord`**. Each MobileTogether design has one **`.mtrecord`** file associated with it. For example, the design file `QS01.mtd` will have a MobileTogether Recording file named `QS01.mtrecord` associated with it. The MT Recording file is saved in the same folder as the design file by clicking the **Save** button of the Manage Test Cases and Runs dialog (*screenshot above*). When you save the MT Recording file subsequently, it overwrites the MT Recording file that already exists.

If you wish to reload test cases and their associated test runs from the MT Recording file of the active design, click **Reload**. In this event, the test cases and test runs in the dialog will be replaced by those in the MT Recording file.

## Deleting and comparing test runs

To delete one or more test cases/runs, select the test cases/runs and click **Delete Selected**. If you select a test case, then all its test runs will also be implicitly selected and deleted; however, you will be warned about this and prompted for a go-ahead. You can also delete all test runs across all test cases by clicking Delete All PLaybacks.

If a test run is different than its test case, then the part that is different is indicated in the *Compared* pane by a pink button (*see screenshot above*). For example, if an action of a test run is different, then the test run's *Actions* cell will contain a pink button (*see screenshot above*). If you click a pink button, the Compare Test Cases and Runs dialog[1414] will be displayed, in which the details of that test run and its test case are shown. To open this dialog, you can also select the test run and its test case, and click **Compare Selected**.

*Properties of a test run*

To see the properties of a test run, select it and click **Compare Selected**. The properties of the test run will be displayed in the Compare Test Cases and Runs dialog[1414].

## Converting an associated test run to a test case

To convert a test run to a test case, select it and click **Replace Reference**. The selected test run will replace its originating test case, and will take the name of the test case. All other test runs that were associated with the replaced test case will be deleted. A warning message will be displayed before the action is carried out, so you can cancel if you wish.

## Server deployment

You can deploy one or more test cases of the active design to the server. You can do this in one of the following ways:

- Deploy the design and its test cases via the **File | Deploy to MobileTogether Server** command.
- Deploy test cases only (without the design) if the design has been deployed already. Do this by clicking **Deploy** in the Manage Test Cases and Runs dialog (*screenshot above*). If you click Browse in this dialog, the deployed solutions are displayed together with the number of already deployed test

cases in round brackets. (The number on angular brackets is the MobileTogether Designer version that was used to generate the test case.)

After a test case has been deployed to the server and been made active, it can be played back each time the solution is started on the client. If multiple test cases have been made active on the server, then all these test cases will be played back on the client. Playback results are stored on the server. You can retrieve these playbacks to MobileTogether Designer by clicking **Retrieve** in the Manage Test Cases and Runs dialog (*screenshot above*). The retrieved playbacks are stored as test runs of the test case that was deployed, and can be handled in exactly the same way as other test runs.

You can delete deployed test cases from the server. Do this as follows: In the Manage Test Cases and Runs dialog (*screenshot above*), select the test case to delete, then click **Delete Deployed**.

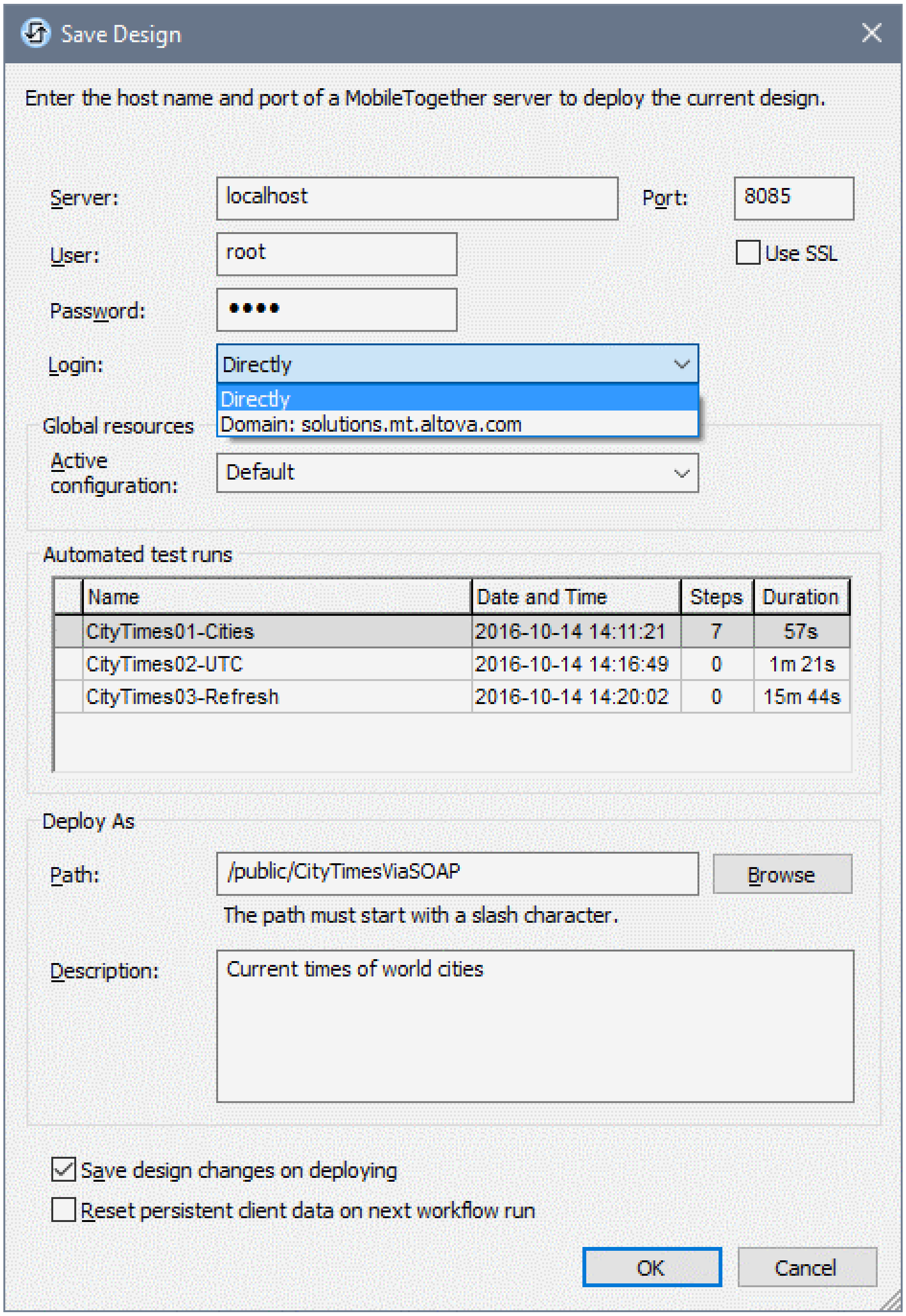
For more details about these procedures, see Deploying to Server[1410].

# 20.4 Deploying Test Cases to Server

You can deploy one or more test cases of the active design to the server. If a test case is made active on the server, then it can be played back each time the solution is started on a client. In this way, a test case can be played back on multiple clients. These playbacks are stored on the server, and can be retrieved in MobileTogether Designer for comparison with the originating test case.

## Deploying test cases to MobileTogether Server

A test case can be deployed with a design to MobileTogether Server via the **File | Deploy to MobileTogether Server** 1574 command. This command displays a dialog (*screenshot below*) in which the following are entered: (i) the server's access details, (ii) the path to the design on the server, and (iii) the test cases to deploy with that design (select the test cases that you want to deploy).



Enter the host name and port of a MobileTogether server to deploy the current design.

|  |  |  |  |
|---|---|---|---|
| Server: | localhost | Port: | 8085 |
| User: | root | □ Use SSL | |
| Password: | •••• | | |
| Login: | Directly | | |

Directly
Domain: solutions.mt.altova.com

Global resources

Active configuration: Default

Automated test runs

| Name | Date and Time | Steps | Duration |
|---|---|---|---|
| CityTimes01-Cities | 2016-10-14 14:11:21 | 7 | 57s |
| CityTimes02-UTC | 2016-10-14 14:16:49 | 0 | 1m 21s |
| CityTimes03-Refresh | 2016-10-14 14:20:02 | 0 | 15m 44s |

Deploy As

Path: /public/CityTimesViaSOAP

The path must start with a slash character.

Description: Current times of world cities

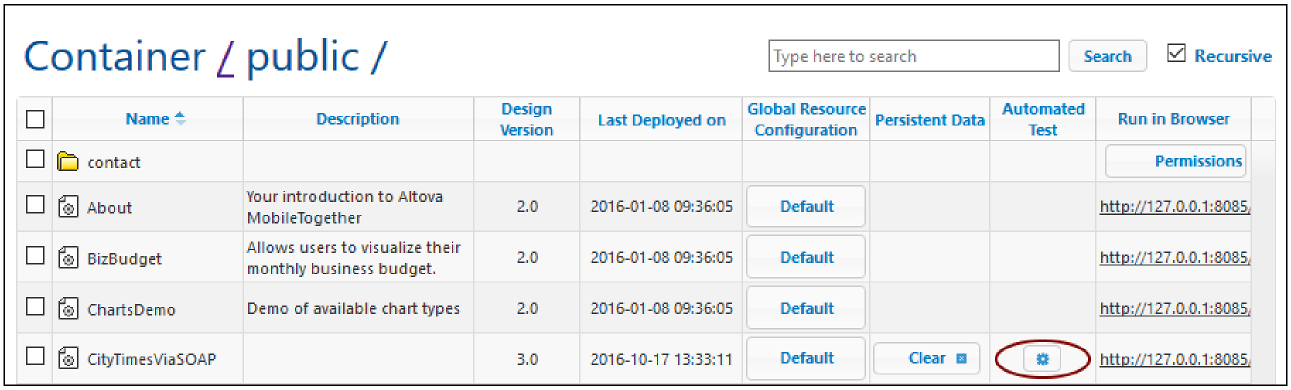
☑ Save design changes on deploying

□ Reset persistent client data on next workflow run

If a design has already been deployed to the server, you can additionally deploy its test cases by clicking **Deploy** in the the Manage Test Cases and Runs dialog[1406]. Test cases are identified by internal IDs, so only test cases with the same ID are overwritten.

## Managing test cases on the server

If a solution on the server has a test case deployed with it, then, in the Workflows tab (*screenshot below*), the solution will have a wheel symbol in its *Automated Test* column (*see screenshot*).

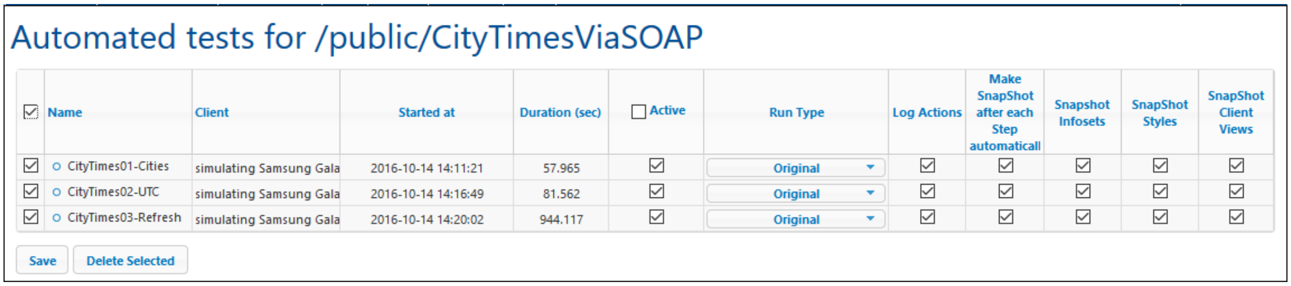


Container / public /

Type here to search | Search | ☑ Recursive

| □ | Name | Description | Design Version | Last Deployed on | Global Resource Configuration | Persistent Data | Automated Test | Run in Browser |
|---|---|---|---|---|---|---|---|---|
| □ | contact | | | | | | | Permissions |
| □ | About | Your introduction to Altova MobileTogether | 2.0 | 2016-01-08 09:36:05 | Default | | | http://127.0.0.1:8085/ |
| □ | BizBudget | Allows users to visualize their monthly business budget. | 2.0 | 2016-01-08 09:36:05 | Default | | | http://127.0.0.1:8085/ |
| □ | ChartsDemo | Demo of available chart types | 2.0 | 2016-01-08 09:36:05 | Default | | | http://127.0.0.1:8085/ |
| □ | CityTimesViaSOAP | | 3.0 | 2016-10-17 13:33:11 | Default | Clear ☒ | ⚙ | http://127.0.0.1:8085/ |

To activate a design's test case and define how the test case is played back on the client, click the solution's wheel icon (*shown in the screenshot above*). This displays a page showing the automated tests of that solution (*screenshot below*).



Automated tests for /public/CityTimesViaSOAP

| ☑ | Name | Client | Started at | Duration (sec) | □ Active | Run Type | Log Actions | Make SnapShot after each Step automaticall | Snapshot Infosets | SnapShot Styles | SnapShot Client Views |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ☑ | ○ CityTimes01-Cities | simulating Samsung Gala | 2016-10-14 14:11:21 | 57.965 | ☑ | Original | ☑ | ☑ | ☑ | ☑ | ☑ |
| ☑ | ○ CityTimes02-UTC | simulating Samsung Gala | 2016-10-14 14:16:49 | 81.562 | ☑ | Original | ☑ | ☑ | ☑ | ☑ | ☑ |
| ☑ | ○ CityTimes03-Refresh | simulating Samsung Gala | 2016-10-14 14:20:02 | 944.117 | ☑ | Original | ☑ | ☑ | ☑ | ☑ | ☑ |

Save | Delete Selected

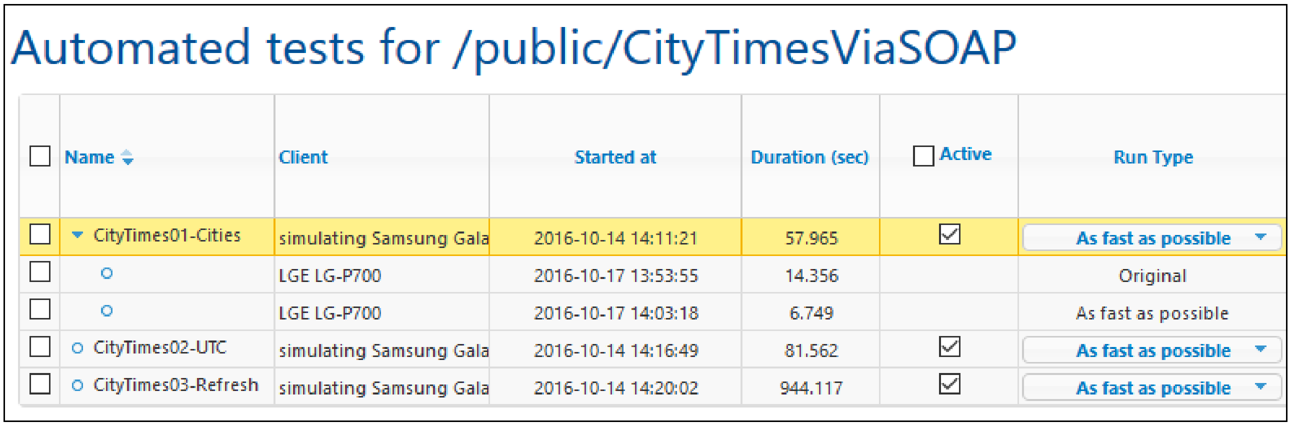
The Automated Tests page shows all the test cases that have been deployed for the selected solution. You can set up individual test cases for playback on client devices as follows:

1. In the *Active* column, check the test cases that you want to make active. These test cases will be played back on the client when the user starts a solution. If multiple test cases are selected, then all the selected test cases will be played back. If any one of a solution's test cases has been activated, then, on the Workflows page, the wheel in the design's *Automated Test* column is displayed in red.
2. Set the logging details you want during playback. Do this by checking the columns you want. See the section *Recording and playback options* in Managing Test Cases and Runs 1406 for information about these options.
3. Click **Save** to finish.
4. In the *Security* and *Devices* tabs, you can specify (i) the users/roles that are allowed to carry out test runs, and (ii) the devices on which to carry out test runs. Select the user/role or the device, and click **Assign** to allow the selected user/role or device.

If you wish to delete a test case, select its check box in the leftmost column and click **Delete Selected**. You can also delete a test case on the server by selecting it in the Manage Test Cases and Runs dialog 1406 (of MobileTogether Designer) and clicking **Delete Deployed**.

## After playback on a client

After a test case is played back on a client, the playback results are stored on the server as a test run and displayed on the Automated Tests page (*screenshot below*). Each playback result is shown as a descendant of its test case. For example, in the screenshot below, it can be seen that two test runs  have been stored for the test case `CityTimes01-Cities`. Notice also that the client device on which playback occurred and the time of playback are also given. To delete a test run, select its check box in the leftmost column and click **Delete Selected**.



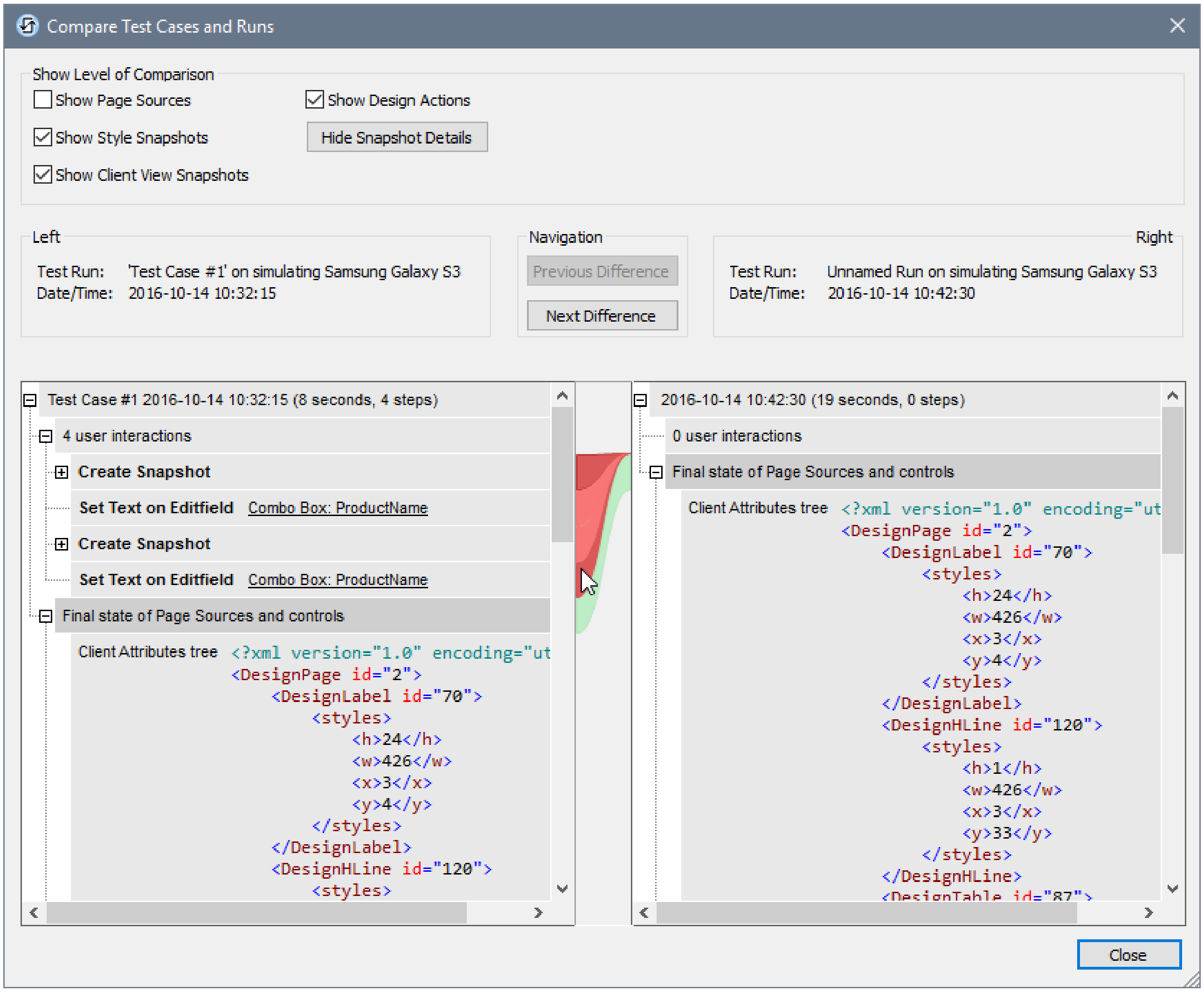
Automated tests for /public/CityTimesViaSOAP

| □ | Name | Client | Started at | Duration (sec) | □ Active | Run Type |
|---|---|---|---|---|---|---|
| □ | ▾ CityTimes01-Cities | simulating Samsung Gala | 2016-10-14 14:11:21 | 57.965 | ☑ | As fast as possible ▾ |
| □ | ○ | LGE LG-P700 | 2016-10-17 13:53:55 | 14.356 | | Original |
| □ | ○ | LGE LG-P700 | 2016-10-17 14:03:18 | 6.749 | | As fast as possible |
| □ | ○ CityTimes02-UTC | simulating Samsung Gala | 2016-10-14 14:16:49 | 81.562 | ☑ | As fast as possible ▾ |
| □ | ○ CityTimes03-Refresh | simulating Samsung Gala | 2016-10-14 14:20:02 | 944.117 | ☑ | As fast as possible ▾ |

You can retrieve test runs to MobileTogether Designer by clicking **Retrieve** in the Manage Test Cases and Runs dialog 1406 (of MobileTogether Designer). Retrieval applies to all test runs of all the deployed test cases of that design, and does not entail the removal of test runs from the server. The retrieved test runs are displayed hierarchically under their originating test cases in MobileTogether Designer and can be handled in exactly the same way as every other test run. For example, you can compare a retrieved test run with its originating test case.

## 20.5 Comparing Test Runs

You can compare a test run with its originating test case to see differences in their steps, actions, page source data, styles, and client views (the layout coordinates of design components in specific clients). In the Manage Test Cases and Runs dialog 1406, select the two test runs that you want to compare, and click **Compare Checked**. The Compare Test Cases and Runs dialog (*screenshot below*) is displayed.



☐ Show Page Sources
☑ Show Style Snapshots
☑ Show Client View Snapshots
☑ Show Design Actions

Left
Test Run: 'Test Case #1' on simulating Samsung Galaxy S3
Date/Time: 2016-10-14 10:32:15

Right
Test Run: Unnamed Run on simulating Samsung Galaxy S3
Date/Time: 2016-10-14 10:42:30

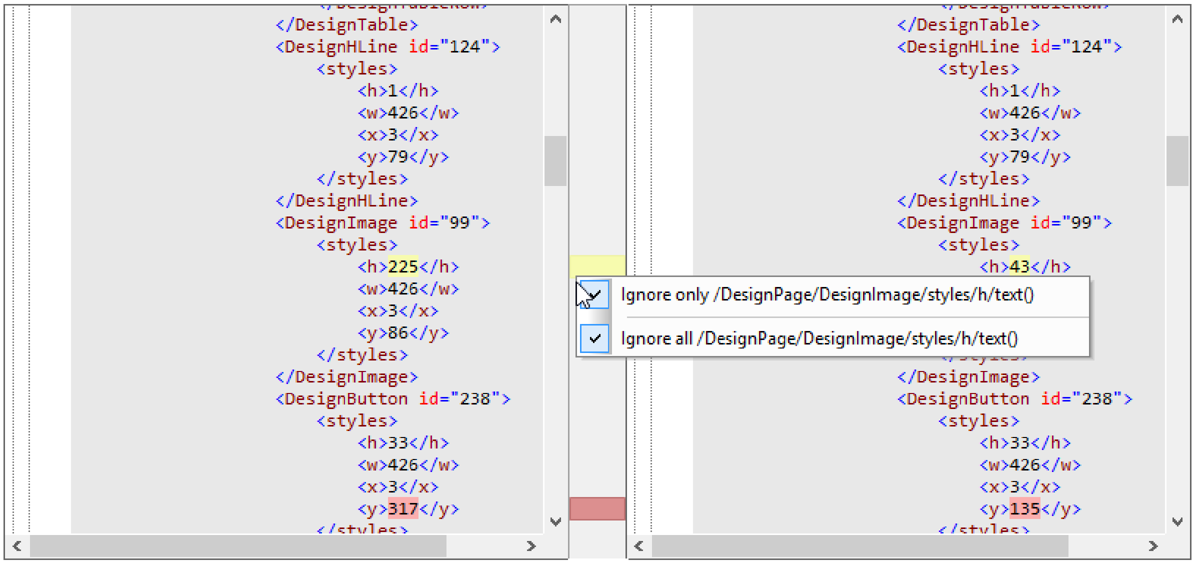
```
⊟ Test Case #1 2016-10-14 10:32:15 (8 seconds, 4 steps)
  ⊟ 4 user interactions
    ⊞ Create Snapshot
      Set Text on Editfield   Combo Box: ProductName
    ⊞ Create Snapshot
      Set Text on Editfield   Combo Box: ProductName
  ⊟ Final state of Page Sources and controls
      Client Attributes tree  <?xml version="1.0" encoding="ut
                              <DesignPage id="2">
                                  <DesignLabel id="70">
                                      <styles>
                                          <h>24</h>
                                          <w>426</w>
                                          <x>3</x>
                                          <y>4</y>
                                      </styles>
                                  </DesignLabel>
                                  <DesignHLine id="120">
                                      <styles>
```

```
⊟ 2016-10-14 10:42:30 (19 seconds, 0 steps)
    0 user interactions
  ⊟ Final state of Page Sources and controls
      Client Attributes tree  <?xml version="1.0" encoding="ut
                              <DesignPage id="2">
                                  <DesignLabel id="70">
                                      <styles>
                                          <h>24</h>
                                          <w>426</w>
                                          <x>3</x>
                                          <y>4</y>
                                      </styles>
                                  </DesignLabel>
                                  <DesignHLine id="120">
                                      <styles>
                                          <h>1</h>
                                          <w>426</w>
                                          <x>3</x>
                                          <y>33</y>
                                      </styles>
                                  </DesignHLine>
                                  <DesignTable id="87">
```

Each test run is displayed in a separate pane as a tree (*see screenshot above*), with their details displayed above the pane. Within the pane, each tree is structured as a chronological sequence of user actions (steps) and design actions. If a snapshot was taken at some point during the test run (for example, after a user action), then the snapshot's details are shown at that point. Each snapshot consists of the following: (i) page source data, (ii) style information, and (iii) layout coordinates of design components on the client.

In the Compare Test Cases and Runs dialog, you can do the following:

- Show/hide the following comparison levels: (i) design actions, (ii) page source data, (iii) styles, and (iv) the layout coordinates of design components on clients. To show/hide one of these levels, check/uncheck its check box respectively.
- Navigate the document by dragging the vertical scroll bars or clicking the **Previous Difference** and **Next Difference** buttons.
- To locate the correspondence of an item in the other test run, place the cursor over the central column at the level of that item. The correspondence is shown by a green connector between the panes, with differences shown by dark red connectors (*see screenshot above*).
- Ignore specified page source or snapshot data nodes for differences. In the column between the two panes, right-click a difference connector and select **Ignore only <*this node*>** or **Ignore all <*these nodes*>** *(see screenshot below)*. Ignored nodes are highlighted in yellow. These nodes will be ignored as differences in future playbacks. To consider differences in these nodes again, deselect **Ignore only <*this node*>** or **Ignore all <*these nodes*>**.



```
</DesignTableRow>
</DesignTable>
<DesignHLine id="124">
    <styles>
        <h>1</h>
        <w>426</w>
        <x>3</x>
        <y>79</y>
    </styles>
</DesignHLine>
<DesignImage id="99">
    <styles>
        <h>225</h>
        <w>426</w>
        <x>3</x>
        <y>86</y>
    </styles>
</DesignImage>
<DesignButton id="238">
    <styles>
        <h>33</h>
        <w>426</w>
        <x>3</x>
        <y>317</y>
    </styles>
```

```
</DesignTableRow>
</DesignTable>
<DesignHLine id="124">
    <styles>
        <h>1</h>
        <w>426</w>
        <x>3</x>
        <y>79</y>
    </styles>
</DesignHLine>
<DesignImage id="99">
    <styles>
        <h>43</h>
    </styles>
</DesignImage>
<DesignButton id="238">
    <styles>
        <h>33</h>
        <w>426</w>
        <x>3</x>
        <y>135</y>
    </styles>
```

☑ Ignore only /DesignPage/DesignImage/styles/h/text()
☑ Ignore all /DesignPage/DesignImage/styles/h/text()

- The last item in both trees is the respective final state. If there is no difference between the final states, then the connector between them will be green; otherwise it will be red.

# 21 Offline Usage

Your MobileTogether solution, whether accessed using the MobileTogether Client app on a client device or an app store app[1471], can be designed to enable a user to work offline (that is, with no active network or Internet connection). Data can be synchronized with one or more databases on the MobileTogether Server when the user reconnects to the server.

This topic: (i) lists the main settings that enable offline work; and (ii) describes how each of these settings affect offline work and can be used in a design.

The sub-topics of this section contain descriptions of different ways in which users can work offline and synchronize subsequently, when they come online.

## Steps to enable offline work

To enable the user to work offline, use one or more of the following settings according to what your design needs are.

- In the Project Settings[296], set Server Access[296] to *On demand*. This ensures that the server will be contacted (for downloading or uploading data) only when the solution initiates such an action.
- In the context menu of page source trees[359], set both Load Data[359] and Save Data[359] to *Not Automatically*.
- Use the **$PERSISTENT** page source[349] to store data on the client.
- If you want to download data to the client, use a Reload action[801].
- Use a Save action[803] or DB Execute action[861] to upload data to the server.

The steps above are general indicators. For a better understanding, see the discussion of these settings below.

For a description of some scenarios of offline use, see the examples in the sub-topics of this section.

## Settings that enable offline work

The following settings determine when and how data is transferred between client and server. These settings effectively configure offline usage and data synchronization. Set their values according to the needs of your solution.

*Project settings*
The Server Access[296] setting is a key setting for determining offline use. It takes one of three values: (i) *Always*, (ii) *On Demand*, (iii) *Never*. The main difference between *Always* and *On Demand* is that in the case of *Always* data transfer between server and client occurs continually, whereas in the case of *On Demand* data transfers occur only when explicitly specified via an action. One consequence of using *On Demand* is that data trees are not automatically updated at solution start. This difference can be seen in the example solutions **02-DisplayRecords.mtd**[1420] and **03-DisplayOnDemand.mtd**[1420], where Server Access has been set, respectively, to *Always* and *On Demand*.

*Load data*
The Load Data[359] setting is available on the root node[349] of page sources. It is not available on **$PERSISTENT** page sources. Load Data[359] takes one of three values: (i) *On first use*, (ii) *On every page*, (iii) *Not automatically*. The value of the setting can be assigned when the page source is created; it can be changed subsequently via the context menu of the source's root node[359]. If you want to work offline and to ensure that the client does not automatically connect to the server (on solution start or when a new page loads), then set

Load Data 359 to *Not Automatically*. Given this setting, the server will not be contacted for automatic data loading, and the client can stay offline. To load data from the server, you would have to implement an explicit action to load data, for example, a Load File 809 action, Reload 801 action, or a DB Execute 861 action.

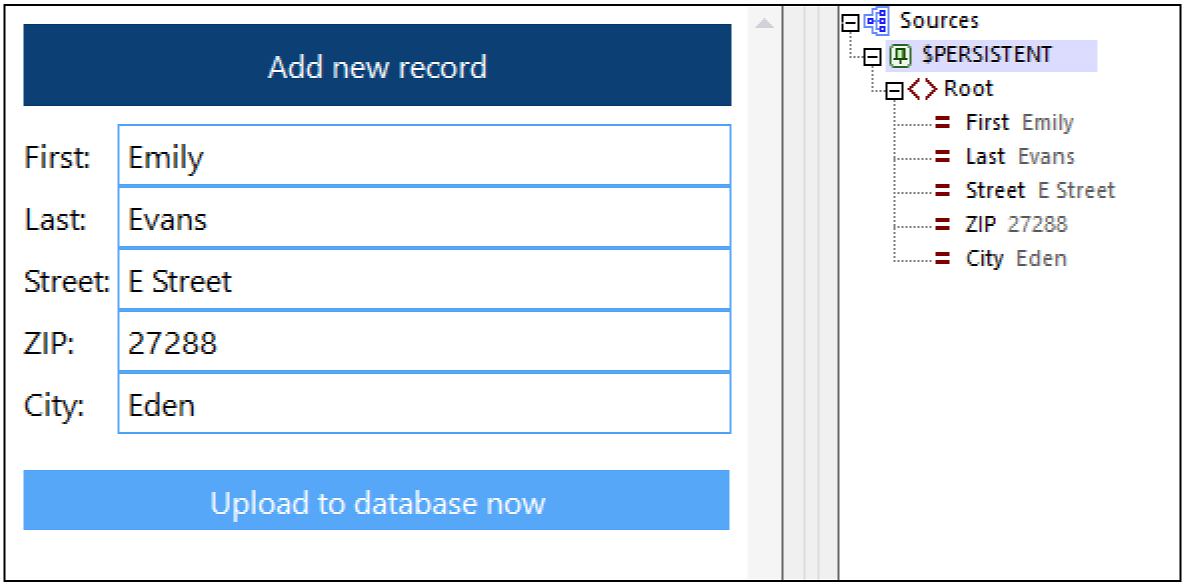
*Save data*
The Save Data 359 setting is available on the root node 349 of any page source that is linked to a data file or a DB. It is not available on **`$PERSISTENT`** page sources. Save Data 359 takes one of four values: (i) *On every page leave*, (ii) *On any solution finish*, (iii) *On last submit,* (iv) *Not automatically*. The value of the setting can be assigned when the page source is created, and it can be changed subsequently via the context menu of the source's root node 359. If you want to work offline and ensure that the client does not automatically connect to the server (on a page leave, a solution finish, or a last submit), then set Save Data 359 to *Not Automatically*. In this case, the server will not be contacted for automatic data saving, and the client can stay offline. To save data to the server, you would have to implement an action such as the Save 801 action.

By setting Server Access 296 to *On Demand* and Load Data 359 and Save Data 359 to *Not Automatically*, you effectively set the client device to be offline.

# 21.1 Enter Data Offline and Upload

The example solution `01-AddRecord.mtd` (*start screen shown below*) is located in the following *(My) Documents* folder: `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\OfflineUsage`. Open the file in MobileTogether Designer and run a simulation (**F5**) to see how it works.

The solution enables records to be entered on a client device, one record at a time, with each being saved to a SQLite database on the server, `Addresses.sqlite`, before the next record is entered. During editing, the record is stored in the **`$PERSISTENT`** page source on the client (*see screenshot below*). The solution's Server Access[296] is restricted by the *On Demand* setting, as a result of which the solution will only communicate with the server when communication is required and is explicitly specified in the solution's workflow. Till such time, the solution is offline and the record's data is entered while the solution is offline. When the user clicks **Upload to database now**, the solution on the client connects with the server to upload the record to the SQLite database on the server.


Add new record

First: Emily

Last: Evans

Street: E Street

ZIP: 27288

City: Eden

Upload to database now

## Key settings

The key settings of the `01-AddRecord.mtd` solution and for working offline are discussed below.

*The Load Data setting*
The Load Data[359] setting is not applicable for this solution because the solution has only one page source, **`$PERSISTENT`**, which is on the client and for which, therefore, no data loading is required. (Since we want to display only the new record in the client—and not all the DB records—no other page source is needed to hold the records of the DB.)

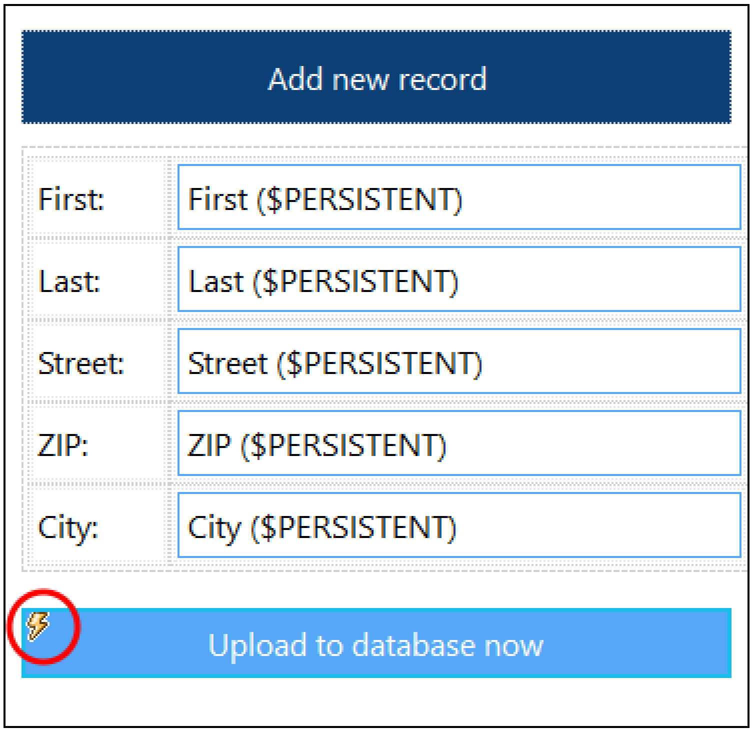
*The Save Data setting*
The Save Data[359] setting is not applicable to **`$PERSISTENT`** page sources.

*The Server Access setting*

The Server Access[296] setting is set to *On Demand*. As a result the server is contacted only when the new record has to be uploaded to the server.

This happens when **Upload to database now** is clicked. This button's `OnButtonClicked` event triggers the following actions:

1. A DB Execute[861] action that executes an SQL statement to insert the data of the **`$PERSISTENT`** tree as a new record on the server DB.
2. An Update Node[886] action to reset the nodes of the **`$PERSISTENT`** tree, each to the empty string. This is required to enable data entry of the next record.

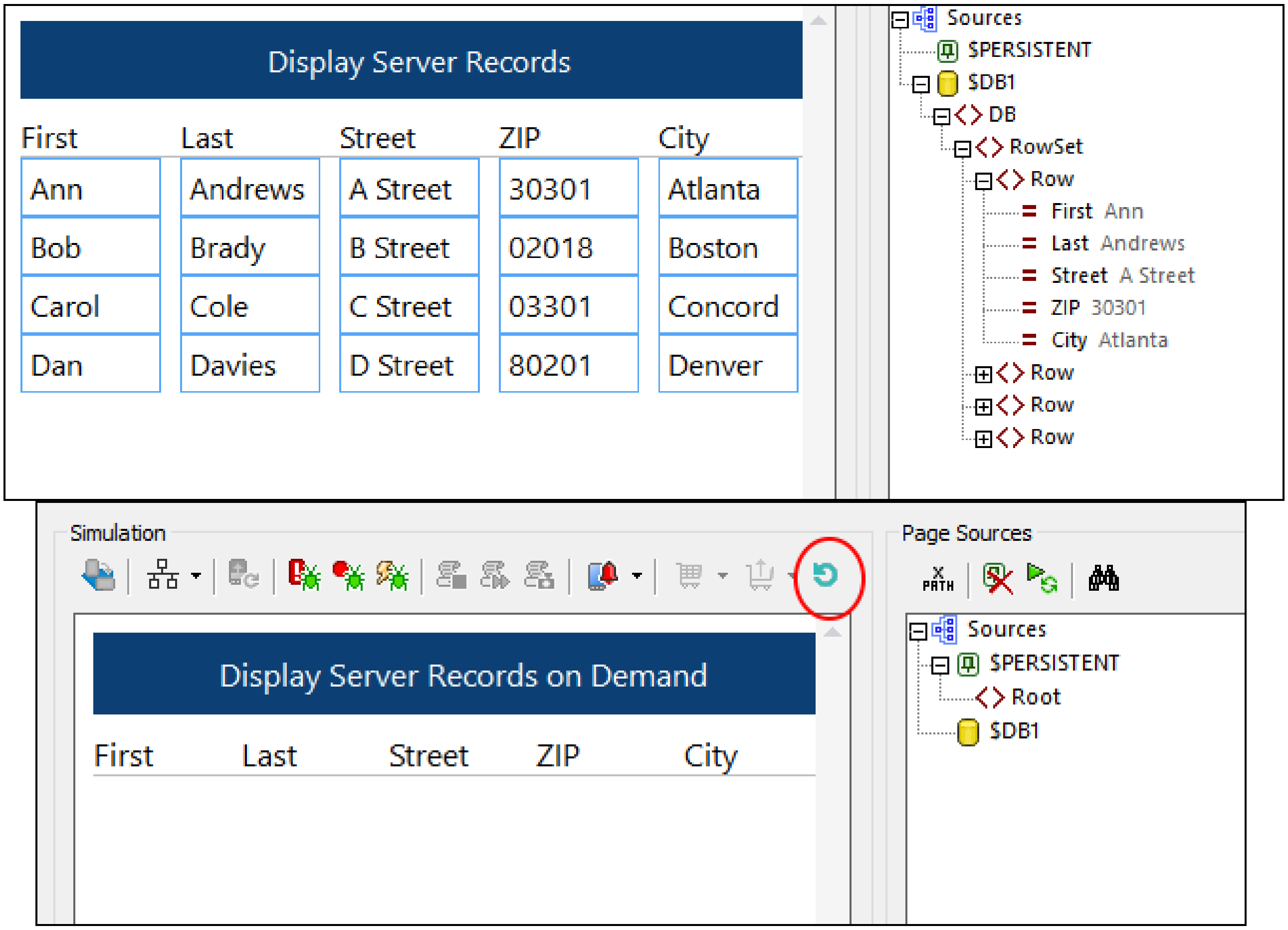


**Note:** In the design, click the Event Actions icon (*circled in red in the screenshot above*) to see how the actions for updating the database (*described above*) have been defined.

# 21.2 Connect to Server on Demand

The example solutions `02-DisplayRecords.mtd` (*start screen shown below left*) and `03-DisplayOnDemand.mtd` (*start screen shown below right*) are located in the following *(My) Documents* folder: `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\OfflineUsage`. Open the files in MobileTogether Designer and run simulations (**F5**) to see how they work.

Both solutions differ from the previous solution, **`01-AddRecord.mtd`**[1423], in that they only display the data of the server-based SQLite database `Addresses.sqlite`; there is no mechanism to add records to the DB. The solutions have been deliberately simplified to focus on the display of data. The difference between the two is in their start screens: While that of **`02-DisplayRecords.mtd` (2)** contains data downloaded from the server, that of **`03-DisplayOnDemand.mtd` (3)** does not display any server data. In **`03-DisplayOnDemand.mtd` (3)**, the solution is offline; it only goes online to download server data when the **Refresh** button (*circled red in screenshot below right*) is clicked.



Display Server Records

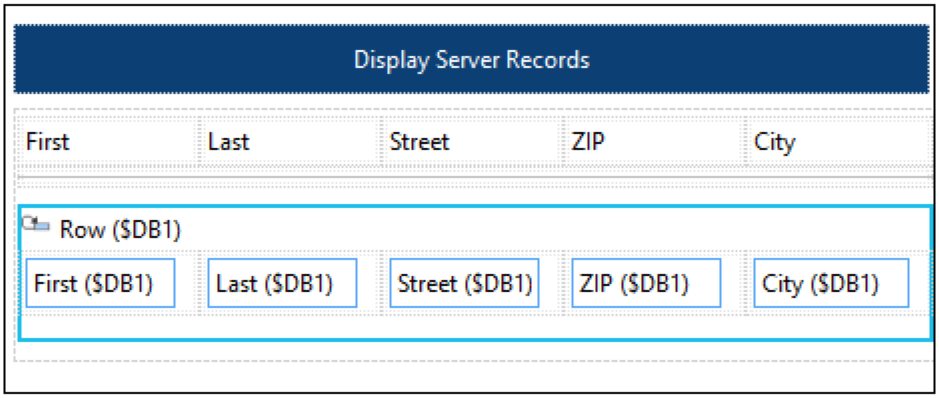
| First | Last | Street | ZIP | City |
|---|---|---|---|---|
| Ann | Andrews | A Street | 30301 | Atlanta |
| Bob | Brady | B Street | 02018 | Boston |
| Carol | Cole | C Street | 03301 | Concord |
| Dan | Davies | D Street | 80201 | Denver |

Display Server Records on Demand

| First | Last | Street | ZIP | City |
|---|---|---|---|---|

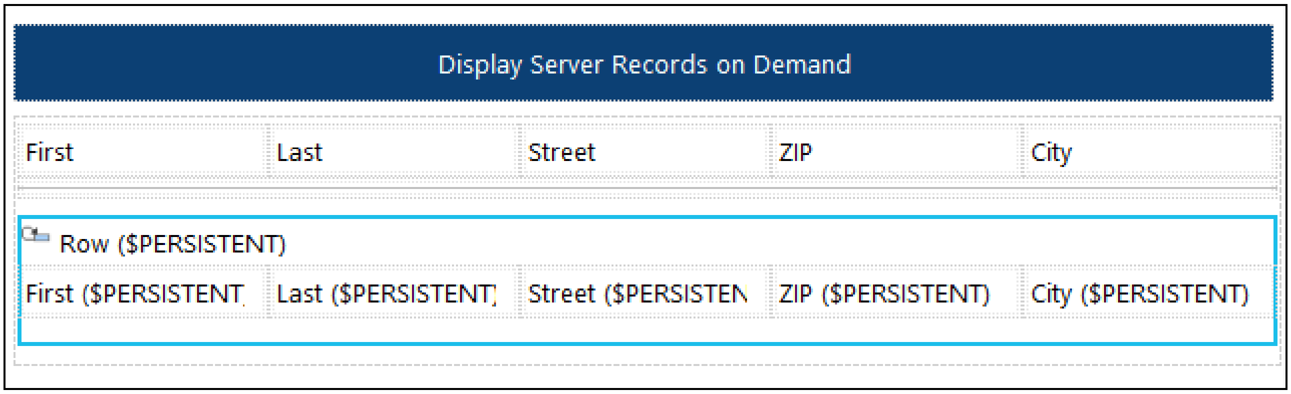
Notice in the screenshots of the designs that the cells of the table in **`02-DisplayRecords.mtd` (2)** are linked to the **`$DB1`** page source (*screenshot below left*), whereas the cells of the table in **`03-DisplayOnDemand.mtd` (3)**, are linked to the **`$PERSISTENT`** page source (*screenshot below right*). These page sources—to **`$DB1`** (the page source linked to the database on the server) and **`$PERSISTENT`** (the page source on the client)—cause data from the respective page sources to be displayed in the table.

- In **(2)**, data is downloaded from the server to **$DB1** on solution start and displayed right away in the start screen.
- In **(3)**, data is downloaded from the server to **$DB1** only when the **Refresh** button is clicked. It is only then that the data is copied to **`$PERSISTENT`** and, because of the page source link in the table, displayed in the table. This is why the table is empty till the **Refresh** button is clicked.



Display Server Records

| First | Last | Street | ZIP | City |
|---|---|---|---|---|
| Row ($DB1) | | | | |
| First ($DB1) | Last ($DB1) | Street ($DB1) | ZIP ($DB1) | City ($DB1) |



Display Server Records on Demand

| First | Last | Street | ZIP | City |
|---|---|---|---|---|
| Row ($PERSISTENT) | | | | |
| First ($PERSISTENT | Last ($PERSISTENT) | Street ($PERSISTEN | ZIP ($PERSISTENT) | City ($PERSISTENT) |

## Key settings

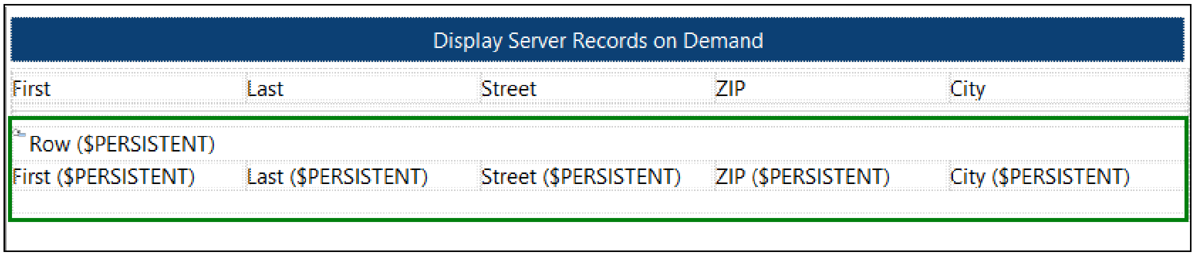
The key settings of both solutions are compared in the following table.

| 02-DisplayRecords.mtd (2) | 03-DisplayOnDemand.mtd (3) | Effect |
|---|---|---|
| Load Data[359] = *On First Use* | Load Data[359] = *Not Automatically* | In **(2)**, DB data is displayed on solution start. In **(3)**, it isn't. |
| Server Access[296] = *Always* | Server Access[296] = *On Demand* | In **(3)**, ensures that the server is contacted only when the **Refresh** button is clicked. |
| No action for OnPageRefresh[390] | OnPageRefresh[390] loads DB data from server | In **(3)**, DB data is loaded from server on page refresh. The **Refresh** button is enabled only when an action is defined for the event. |

*The Load Data setting*
The reason that **(2)** contains downloaded data while **(3)** does not is due to the Load Data 359 setting, which in **(2)** is *On First Use* and in **(3)** is *Not Automatically*. In **(2)** data is loaded when the solution starts.

*The Server Access setting*
The Server Access 296 setting (*Always, On Demand,* or *Never*) does not affect the data download on solution start (which depends on the Load Data 359 setting). But the Server Access 296 setting lets you decide when the server will be contacted. The *On Demand* setting provides you with greater control of the design and the communication process. For example, in **(3)**, the table data is downloaded from the server only when the solution's **Refresh** button is clicked. In the design, this is specified by defining the following set of actions for the `OnPageRefresh` 390 event:

1. Reload 801 the **`$DB1`** page source from the server. The server is contacted and the data is downloaded.
2. Delete the Row nodes 879 of the **`$PERSISTENT`** tree. This is preparatory to adding to **`$PERSISTENT`** the just updated row nodes of **`$DB1`** (*see previous step*). Note that the data is removed only from the **`$PERSISTENT`** page source on the client; the database on the server is not affected.
3. Update the Persistent tree's Root node 886 with the just updated row nodes of the **`$DB1`** tree. This is required because it is the nodes of the **`$PERSISTENT`** page source (and not those of **`$DB1`**) that are displayed in the table (*enclosed in green in screenshot below*).



Display Server Records on Demand

| First | Last | Street | ZIP | City |
|---|---|---|---|---|
| Row ($PERSISTENT) | | | | |
| First ($PERSISTENT) | Last ($PERSISTENT) | Street ($PERSISTENT) | ZIP ($PERSISTENT) | City ($PERSISTENT) |

The point to note is that in **(3)** the server is accessed and data downloaded only when the Reload 801 action is triggered.

## 21.3 Edit Data Offline and Synchronize

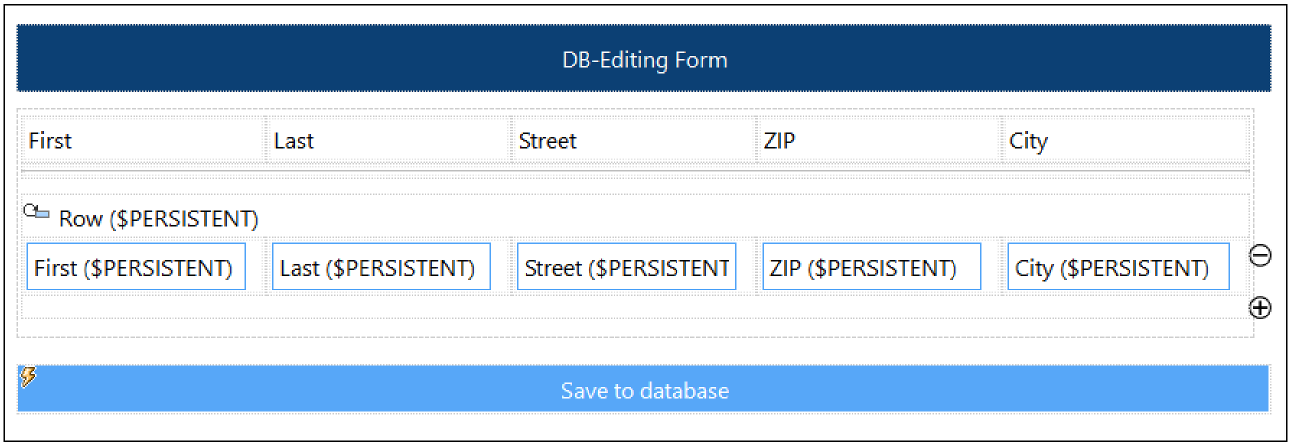
The example solutions `04-EditRecords.mtd` and `05-EditRecordsOnStart.mtd` are examples of how to edit data offline and go online solely to save data to a database on the server.

They are located in the following *(My) Documents* folder: `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\OfflineUsage`. Open the files in MobileTogether Designer and run simulations (**F5**) to see how they work.

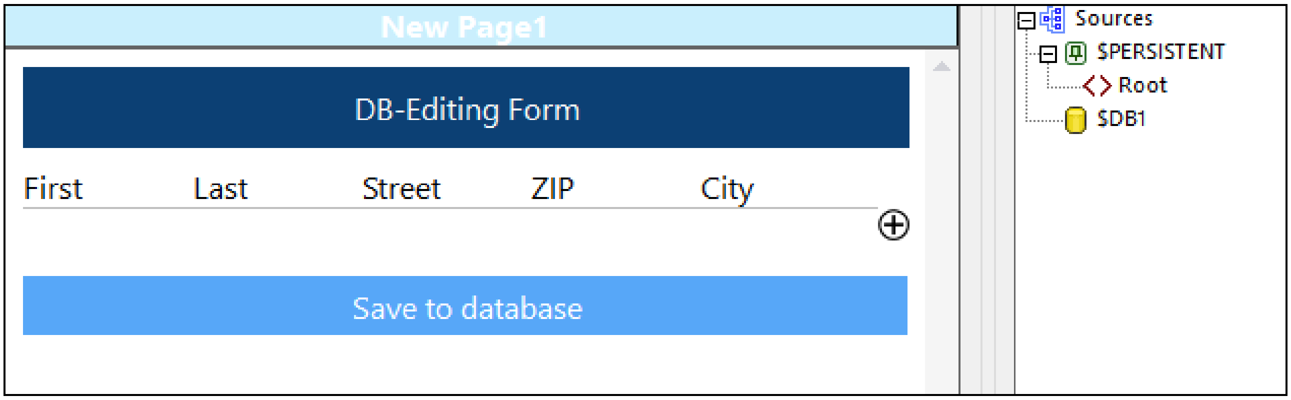
Both solutions save data to a SQLite database on the server, `AddressesIndexed.sqlite`, The solution `04-EditRecords.mtd` is different from `05-EditRecordsOnStart.mtd` in one respect only: that records from the SQLite database *are not displayed* in the start screen of the former solution, but *are displayed* in the start screen of the latter.

### Description of the example solutions

The example solution **`04-EditRecords.mtd`** (*design shown in screenshot below*) displays the DB data in a table and allows users to edit the data. Fields of the table are linked to the **`$PERSISTENT`** page source. This means (i) that it is the contents of **`$PERSISTENT`** that are displayed in the table, and (ii) that edits you make in the table are stored in **`$PERSISTENT`**.

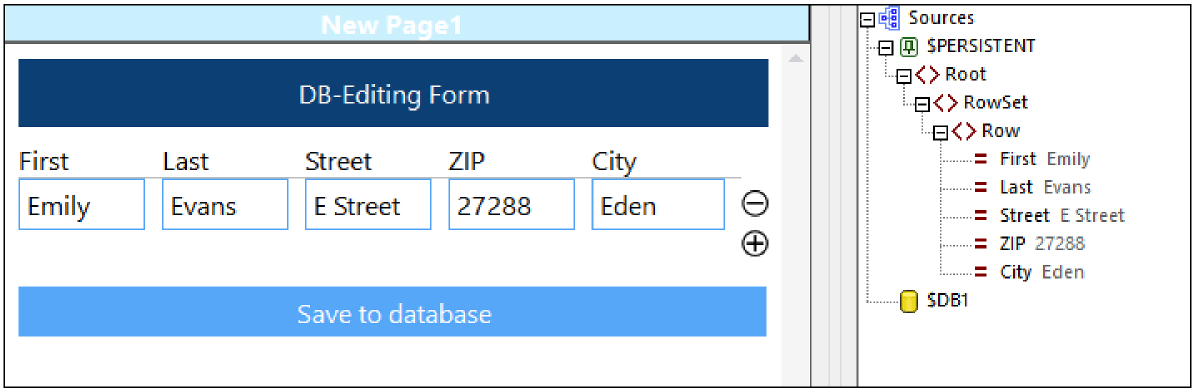


When the solution **`04-EditRecords.mtd`** starts, the table is empty (*see screenshot below*). This is because the table has **`$PERSISTENT`** as its page source and **`$PERSISTENT`** is empty. Note also that the **`$DB1`** page source is empty. This is because it has been set to Load Data = *Not Automatically*. In the Page Sources pane, right-click **`$DB1`** to see the value of its Load Data 359 setting.

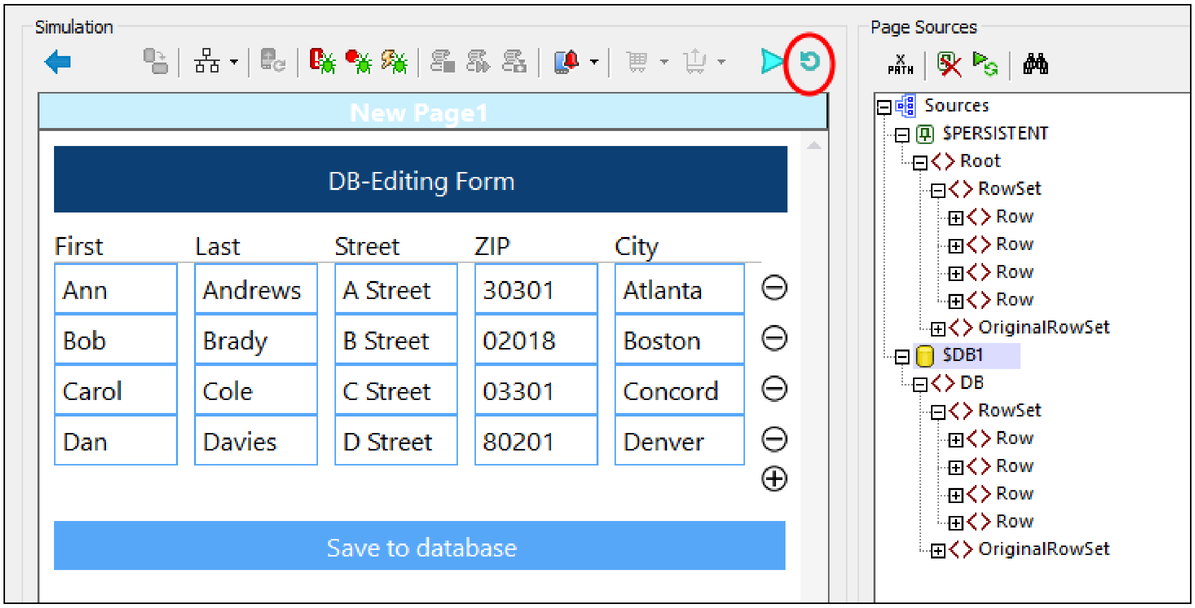


In the solution, the following data edits are possible:

- A new record (row) can be added to the table and the fields of the record can be edited. To add a new row, click the **Plus** icon located to the right of the last row *(see screenshots above and below)*. After you add a new row, the data of the row is stored in **`$PERSISTENT`** (*see screenshot below*).



- In the solution, a row can be deleted by clicking the its **Minus** icon (*see screenshot above*).
- The data in **`$PERSISTENT`** can be saved to the server by clicking **Save to database**. For an explanation of how this works, see the section *Saving edited data by using primary keys and OriginalRowSets* below.
- On clicking **Refresh** (*circled red in screenshot below*), all data rows are downloaded from the server to **`$DB1`** and copied from there to **`$PERSISTENT`** (see the actions of the `OnPageRefresh` 390 event as defined in this event's tab) 390. Any data that was present in **`$PERSISTENT`** before the refresh will be deleted. Since the rows in **`$PERSISTENT`** are displayed in the table, the table now displays all the server records. Note that the Server Access setting for **`$DB1`** has been set to *On Demand*. This ensures that the server will be contacted only when a request is made to the server, such as that for reloading database data. In effect, this means that users can work offline till they need to upload data to the server.



DB-Editing Form

| First | Last | Street | ZIP | City |
|---|---|---|---|---|
| Ann | Andrews | A Street | 30301 | Atlanta |
| Bob | Brady | B Street | 02018 | Boston |
| Carol | Cole | C Street | 03301 | Concord |
| Dan | Davies | D Street | 80201 | Denver |

Save to database

## Key settings

The key settings of the (`04-EditRecords.mtd`) solution are given in the following table.

| 04-EditRecords.mtd | Effect |
|---|---|
| Load Data (of `$DB1`) = *Not Automatically* | Server DB data is not displayed on solution start. |
| Server Access (of `$DB1`) = *On Demand* | The client connects the server only when required by an action. |
| `OnPageRefresh` reloads DB data from server, deletes nodes of `$PERSISTENT`, and appends the new nodes of `$DB1` to `$PERSISTENT`. | On page refresh; `$DB1` is reloaded with data from the server, and this data is copied to `$PERSISTENT`. As a result, the table will contain the latest data on the server. |

## At solution start, load all data for editing

When the solution `04-EditRecords.mtd` is started, the table is empty because **`$PERSISTENT`** is empty. If we want to show all the server DB data in the table, we could use the following strategy, which has been used in **`05-EditRecordsOnStart.mtd`**. To see how this has been set up, open **`05-EditRecordsOnStart.mtd`** and see the definition of the OnPageLoad[390] event of the solution page.

1. Since the actions we want to carry out must occur when the solution starts, we create them on the OnPageLoad[390] event of the solution's Start page.
2. Reload **`$DB1`** via the Reload[801] action. This downloads all DB data from the server to **`$DB1`** (when the solution starts).
3. Delete all the records of **`$PERSISTENT`** via the Delete Node(s)[879] action.
4. Copy all the record nodes from **`$DB1`** to **`$PERSISTENT`**. by using the Update Node(s)[886] action. Once the data has been appended to the **`$PERSISTENT`** tree, it will automatically be displayed in the table.

This is because the controls in the table's cells have page source links to the nodes of the **$PERSISTENT** page source.

## Saving edited data by using primary keys and OriginalRowSets

In both `04-EditRecords.mtd` and `05-EditRecordsOnStart.mtd`, the setting to create an `OriginalRowSet` node has been set to true. This setting is switched on via a toggle command in the context menu of the page source. In both our examples, whenever the page is refreshed (via the **Refresh** button), the `$DB1` page source is reloaded from the server and two important steps are executed: (i) an `OriginalRowSet` node is created in `$DB1` that contains an exact deep copy of the tree's `RowSet` node (that is, of all the rows of the server DB), and (ii) the `RowSet` and `OriginalRowSet` nodes are copied from **$DB1** to the **$PERSISTENT** tree. This way we know what rows were originally present in the table before editing started: edited rows are in `RowSet`, while original rows are in `OriginalRowSet`.

Now if the table is edited, that is, if new rows are added or deleted, then, in **$PERSISTENT**, the number of rows in `RowSet` and `OriginalRowSet` will be different. Each row is identified by a unique primary key field named `ID`, which may not be `NULL` and is automatically set to an auto-incremented integer when the row is added.

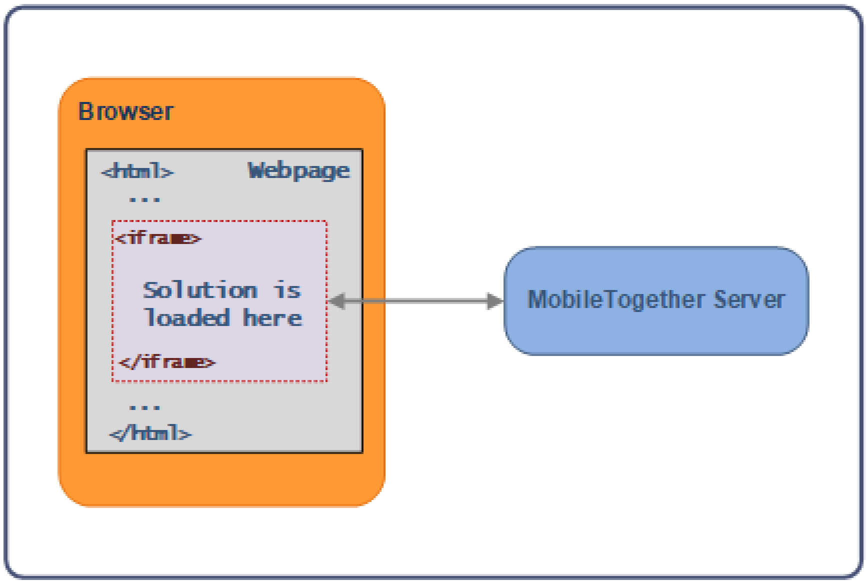
When the data is saved to the server DB, the saving mechanism is as follows:

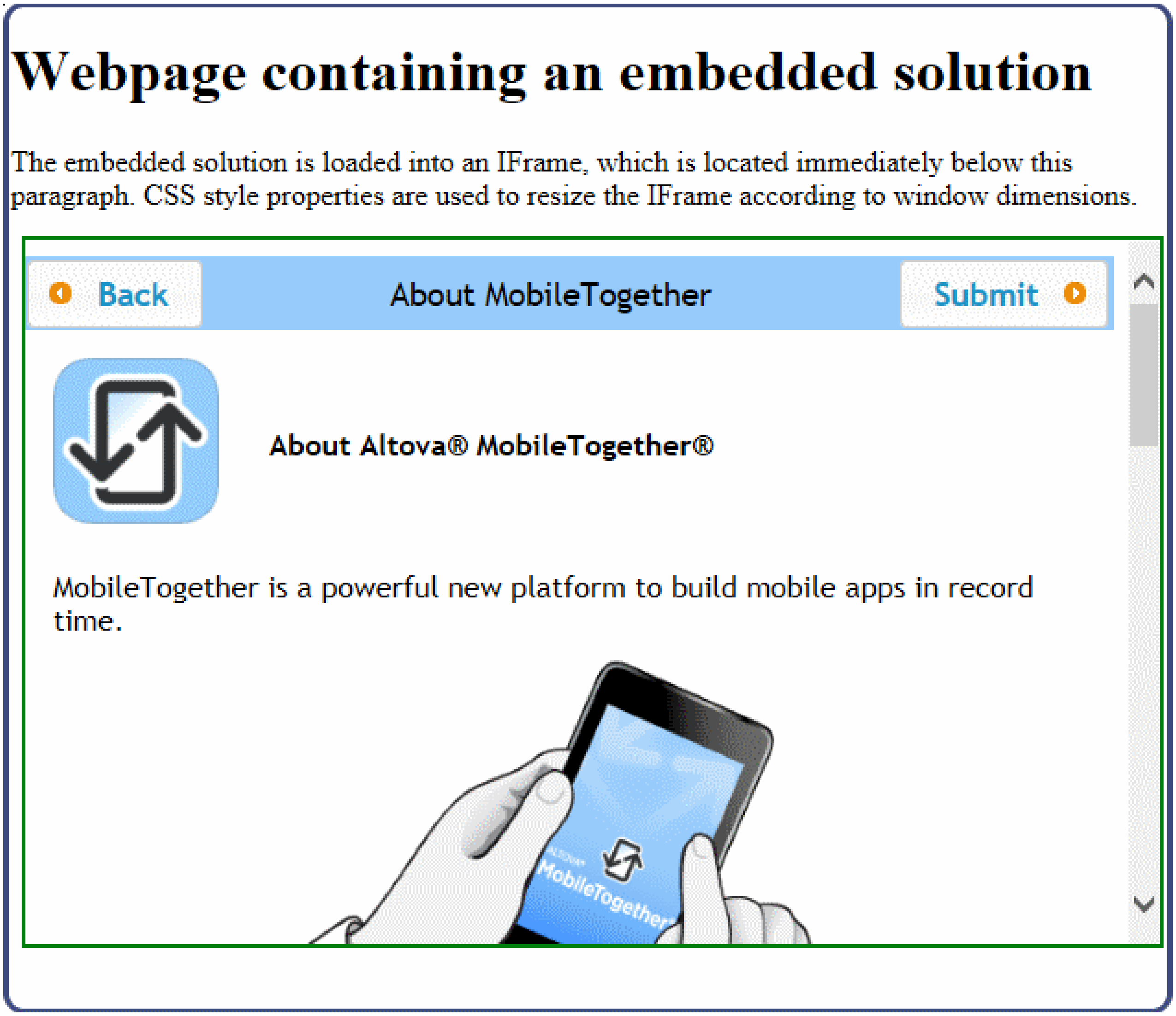
1. **$DB1** is reloaded by using the Reload 801 action.
2. From **$DB1/RowSet**, those rows are deleted that have the same `ID` as that of rows in **$PERSISTENT/OriginalRowset**. This ensures that all the original rows (before editing) are deleted from **$DB1/RowSet**.
3. The rows of **$PERSISTENT/Rowset** are copied to **$DB1/RowSet**. with the Append Node(s) 875 action. These are the new, the edited, and the unedited rows in **$PERSISTENT/Rowset** that we want to save to the server. Rows that have been deleted from the table will not be in this rowset and so will not be copied to **$DB1**. As a result, **$DB1** will now contain only the rows that we want to save back to the server.
4. A Save 803 action saves the data from **$DB1** to the SQLite database on the server, with only modifications being saved.

# 22 Embedded Webpage Solutions

An **embedded webpage solution** is a solution that is embedded in a webpage. In effect, the HTML code of the webpage will contain an **`IFrame`** element, into which the solution is loaded. Data can be exchanged between the webpage and its embedded solution with the use of JavaScript. The solution itself interacts with MobileTogether Server in the usual way, and receives data from MobileTogether Server that can then be communicated back to the webpage via JavaScript mechanisms.

The figure below left shows how the embedded solution interacts with its embedding webpage and MobileTogether Server. The screenshot below right shows a webpage that contains an embedded solution (framed in green).


Browser
<html>
Webpage
...
<iframe>
Solution is loaded here
</iframe>
...
</html>
MobileTogether Server



# Webpage containing an embedded solution

The embedded solution is loaded into an IFrame, which is located immediately below this paragraph. CSS style properties are used to resize the IFrame according to window dimensions.

About MobileTogether

About Altova® MobileTogether®

MobileTogether is a powerful new platform to build mobile apps in record time.

This section is organized as follows:

- Embedding a Solution in a Webpage[1431] describes how to load a solution into an **`IFrame`** element. It contains a complete HTML code listing that you can try out yourself.
- Communication between Webpage and Server[1434]: In addition to embedding the solution in the webpage, we must also enable communication between the different components. The webpage must be able to communicate with the solution in the IFrame. It does this by using JavaScript to post messages to the IFrame and to listen for return messages from the IFrame. Within the IFrame itself, data is communicated between the solution and MobileTogether Server. The section Posting: From Webpage to Solution[1435] describes communication from webpage to IFrame to MobileTogether Server, while the section Listening: From Solution to Webpage[1436] explains how to communicate from MobileTogether Server to IFrame to webpage.
- Authentication[1438]: Any communication that tries to access a workflow on MobileTogether Server must be authenticated. This section describes the types of authentication that an embedded solution can use. Authentication via JSON Web Tokens (JWT)[1439] is a type of authentication that is specific to

embedded solutions; it enables these solutions to be conveniently integrated into existing networks and systems.

- Examples 1445: This section contains the code listings of HTML pages that use embedded webpage solutions, together with step-by-step descriptions of the communication process between webpage and server.

## Useful design mechanisms

The following design mechanisms provide crucial functionality for the Embedded Webpage Solutions feature:

- The **`OnEmbeddedMessage`** 397 page event of the solution picks up the message from the webpage
- The **`$MT_EMBEDDEDMESSAGE`** 397 JSON page source stores the received data in a structured form
- The Load from String 829 action parses a serialized string and places the deserialized structure in a page source; useful for deserializing an XML string in a JSON node and creating an XML page source
- The Save to String 829 action serializes a page source and places the resultant string in the node of another page source
- The Embedded Message Back 924 action sends a serialized JSON string to the IFrame that loaded the current solution

# 22.1 Embedding a Solution in a Webpage

You can embed one or more MobileTogether solutions in a webpage. Each solution is embedded in an HTML `IFrame` element. To embed, do the following:

1. Add the HTML `IFrame` element at the location in the webpage where you want to display the solution (*see example below*)
2. Set the `src` attribute of `IFrame` to the URL of the solution on MobileTogether Server that you want to embed (*see listing below*)

```
<iframe src="http://localhost:8083/run?d=/public/my-mt-solution"
frameborder="0"></iframe>
```

When the HTML page is loaded, its `IFrame` element will load the targeted solution. Since the IFrame will access MobileTogether Server, three authentication-related scenarios are possible:

- *Anonymous access*[1438]*:* If anonymous access to the solution is enabled on MobileTogether Server, then the solution will be displayed directly in the IFrame; no authentication of the user is required. To set anonymous access, the server needs to (i) allow anonymous login (*see the server settings for mobile client ports*), and (ii) permit anonymous use of the solution (by setting the permissions of the workflow's container to a minimum of `container=read` and `workflow=read,use`).
- *User login*[1434]*:* If anonymous user login is not enabled on the server, then the user will be prompted to provide valid MobileTogether Server login credentials. If the user has permission to access the targeted solution, then the solution will be downloaded from MobileTogether Server to the IFrame of the webpage.
- *JWT authentication*[1439]*:* The user is authenticated in a system outside the MobileTogether Server authentication system, and the authentication information is passed to MobileTogether Server via JSON Web Tokens (JWT). JWT authentication enables the targeted solution to load without requiring MobileTogether-specific authentication.

## Example: Simple webpage containing an embedded solution

The HTML code listed below shows how an **`IFrame`** element is used to embed a solution. Below the listing is a screenshot of the HTML page containing the embedded solution (framed in green). The solution used in this listing is a sample named `About` that is packaged with MobileTogether Server; it is located by default in the server's **`public`** container. In order to correctly embed this solution, set up the server to allow anonymous access[1438] of the `About` workflow. You can try out this feature by copying the HTML code below and saving it to file, and then opening the file in a browser.

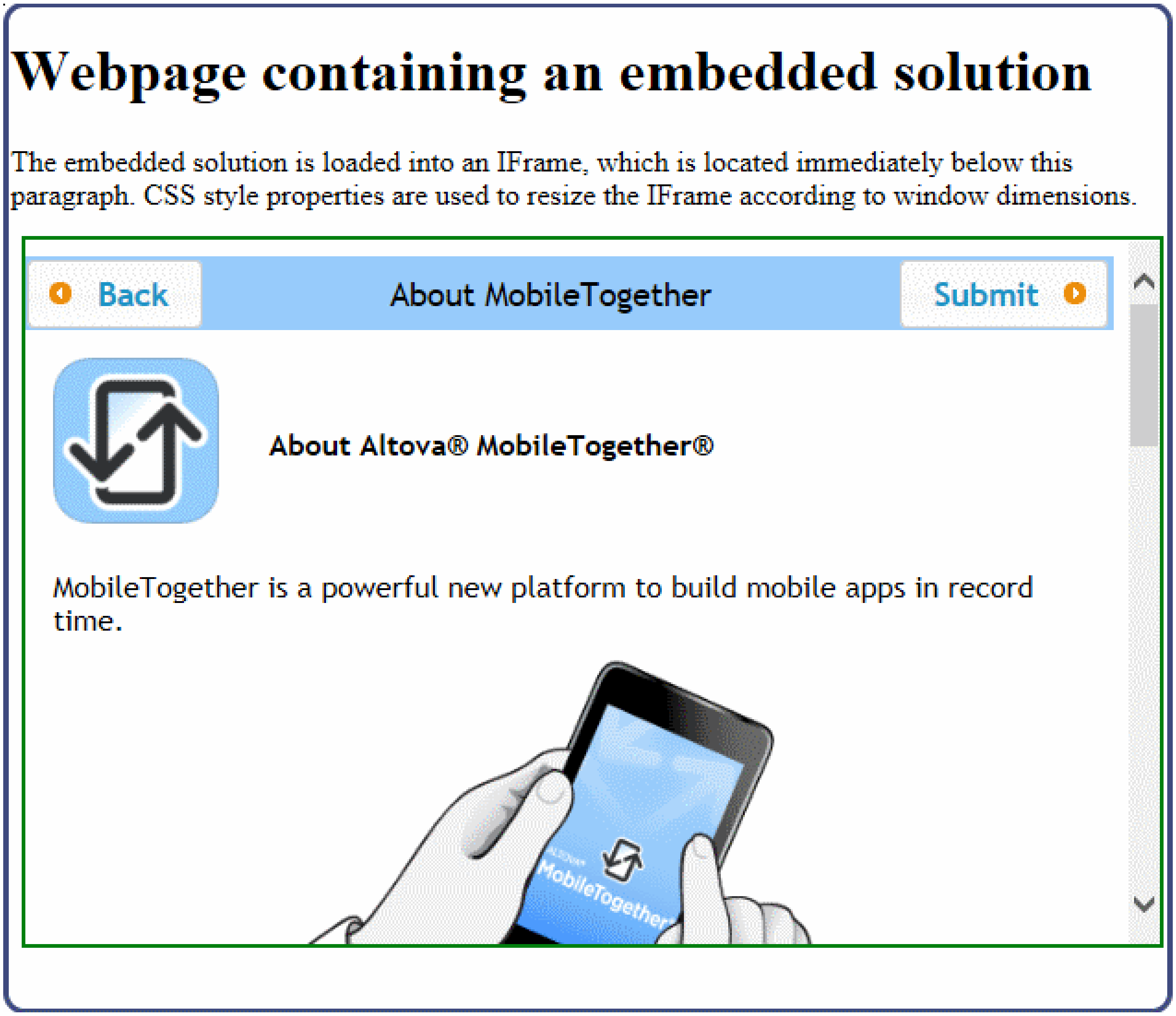
```
<!DOCTYPE html>
<html>
  <head>
    <title>Webpage containing an embedded solution</title>
  </head>
  <body>
    <h1>Webpage containing an embedded solution</h1>
    <p>The embedded solution is loaded into an IFrame, which is located immediately below
this paragraph.</p>
    <div class="resize">
      <iframe src="http://localhost:8083/run?d=/public/About" frameborder="0"></iframe>
    </div>
  </body>
</html>
```

The URL in the **src** attribute must resolve to the following pattern:

```
http://<serveraddress>:<serverport-for-client>/run?d=/<path-to-container>/<solution-name>
```

*Complete listing (containing CSS styles to resize IFrame)*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Webpage containing an embedded solution</title>
    <style>
      .resize {
        position: relative;
        padding-bottom: 56.25%; /* proportion value to aspect ratio 16:9 (9/16 = 0.5625 or 56.25%) */
        padding-top: 30px;
        height: auto;           /* alternatively, try a value of 0 */
        overflow: hidden;
      }

      .resize iframe {
        position: absolute;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
      }
    </style>
  </head>
  <body>
    <h1>Webpage containing an embedded solution</h1>
    <p>The embedded solution is loaded into an IFrame, which is located immediately below this paragraph. CSS style properties are used to resize the IFrame according to window dimensions so that the IFrame's aspect ratio is maintained at 16:9.</p>
    <div class="resize">
      <iframe src="http://localhost:8083/run?d=/public/About" frameborder="0"></iframe>
    </div>
  </body>
</html>
```



# Webpage containing an embedded solution

The embedded solution is loaded into an IFrame, which is located immediately below this paragraph. CSS style properties are used to resize the IFrame according to window dimensions.

**About Altova® MobileTogether®**

MobileTogether is a powerful new platform to build mobile apps in record time.

# 22.2 Communication between Webpage and Server

One reason that a solution would be embedded in a webpage is to exchange data between the webpage and the embedded solution so that the solution processes the input data via MobileTogether Server and returns the result to the webpage. A typical scenario would be the following:

1. A user fills data in an **HTML webpage** form
2. This data is communicated to the **solution** (that is currently loaded in an **IFrame** of the webpage)
3. The solution sends the data to the solution's **workflow on MobileTogether Server**, where it is processed in the usual way
4. Results are returned to the IFrame, where they can be (i) displayed as part of the solution, and/or (ii) passed back to the webpage for display or further processing

**Note:** In the description of this feature, we distinguish between the term *solution* (the display in the IFrame) and *workflow* (the design that is deployed on the server).

The entire round trip consists of the following stages: webpage–solution–workflow–solution–webpage. The mechanisms that are used to send data between webpage and workflow are described in the sub-sections of this section:

- Posting: From Webpage to Server 1435
- Listening: From Server to Webpage 1436

## Data transfer mechanisms

Data is transferred between webpage and server in two stages: webpage–solution and solution–workflow. The two stages use the following mechanisms, respectively:

*Webpage–solution*
Communication between the webpage and the solution is done with JavaScript:

- The **`Window.postMessage()`** method is used to send data from the webpage to the embedded IFrame. (The message is automatically sent onward from the solution to the workflow.)
- The **`Window.addEventListener()`** method is used inside the webpage to listen for a message event that is sent by the workflow to the IFrame. When a message is received by the IFrame, it is forwarded to the webpage, where a JavaScript function can be used to process the message and display it in the webpage

Both the methods listed above are W3C specifications. For more information about them, see the descriptions at the Mozilla Developer Network: **`PostMessage`** and **`AddEventListener`**. Also, since the event listener listens for a message event, see **`MessageEvent`**.

*Solution–workflow*
Communication between the solution and the workflow is based on the fact that the data is accessed in the workflow from a JSON page source (named `$MT_EMBEDDEDMESSAGE`). Because the JSON page source must be created from a JSON structure, the solution does the following: (i) It automatically serializes that message that is received in the IFrame into a JSON string, and (ii) It automatically sends the serialized JSON string to the workflow (where it can be created as the `$MT_EMBEDDEDMESSAGE` JSON page source).

When the communication occurs in the reverse direction (from workflow to solution), it is sent as a JSON string (typically by serializing the `$MT_EMBEDDEDMESSAGE` JSON page source).

## 22.2.1 Posting: From Webpage to Server

A message that is sent from a webpage (via an IFrame embedded in that webpage) to the server is called an **embedded message**. It is sent in the following stages:

1. JavaScript is used to send the embedded message from the webpage to the IFrame. The embedded message is sent as a JSON object that is the first parameter of the `postMessage()` method.
2. When the embedded message reaches the IFrame, the solution serializes the message to a JSON string and sends this string to the solution's workflow on MobileTogether Server. This step is carried out automatically; you do not need to specify any processing.
3. In the workflow, the embedded message can be picked up with the **OnEmbeddedMessage**[397] event. If an action is specified for this event, then the `$MT_EMBEDDEDMESSAGE` JSON page source is created automatically. You can define additional actions as desired.

In this section, we discuss the `postMessage()` method of Step 1. See the HTML webpage examples that respectively post JSON[1445] and XML[1453] data for a detailed description of the whole process. Step 2 is carried out automatically, and so requires no explanation. Step 3 is described in detail in the JSON[1445] and XML[1453] examples. Also see the description of the **OnEmbeddedMessage**[397] event for more information.

### About the postMessage method

Note the following key points:

- The embedded message is passed as the first parameter of `postMessage()`
- The embedded message is passed as a JSON object
- The data in the message is serialized using the structured clone algorithm. This means that the message can contain any of a range of data objects; serialization will be carried out automatically. Note, however, that not all data types can be sent to MobileTogether Server

For example, if the data to be sent is in JSON format, then the `postMessage()` method can send the JSON data structure without any modification. So, if a JSON object is assigned to a variable named `myJSONData`, then this data structure can be sent with the `postMessage()` method like this:

```
function sendMyMessage() {
         document.querySelector('iframe').contentWindow.postMessage(myJSONData,
'*');
         }
```

In this example, the JSON data structure will be received by the IFrame. The solution in the IFrame will send the data to the workflow as a JSON string. In the workflow, the data can be picked up by the **OnEmbeddedMessage**[397] page event and stored in the `$MT_EMBEDDEDMESSAGE` JSON page source.

See this Mozilla Developer Network documentation for detailed information about `postMessage()`. Also see the JSON[1445] and XML[1453] examples.

## 22.2.2 Listening: From Server to Webpage

A message is sent from MobileTogether Server to a webpage in the following stages:

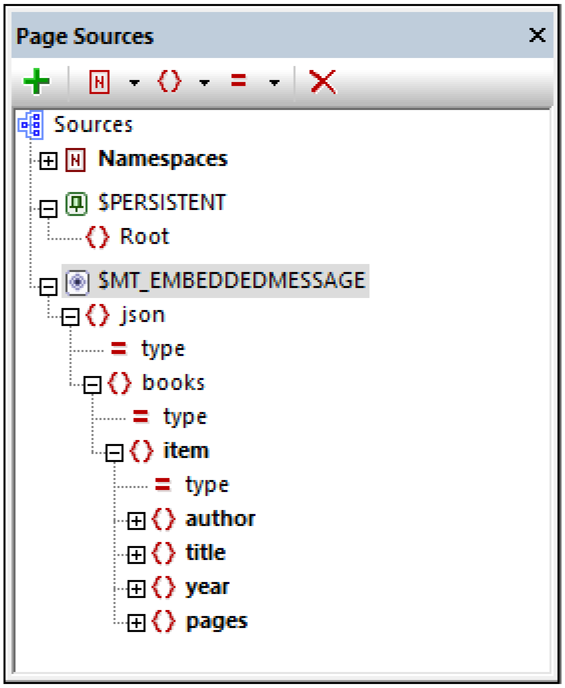
1. The workflow's Embedded Message Back [924] action sends the message to the IFrame in the form of a serialized JSON string. You specify the message to send in the action.
2. When the message reaches the IFrame, it is forwarded to the webpage, where an event listener picks up the **`message`** event and calls a function to process the message. You can register an event listener for the **`message`** event as follows: **`window.addEventListener('message', ProcessReturnMsg)`**
3. The function (`ProcessReturnMsg` in the example above) takes the **`message`** event in the form of a deserialized JSON object as its parameter. You can now access the object as usual and use it in the HTML page. For example:

```
function ProcessReturnMsg(m) {
msgVar = m.data.json.books
...
}
```

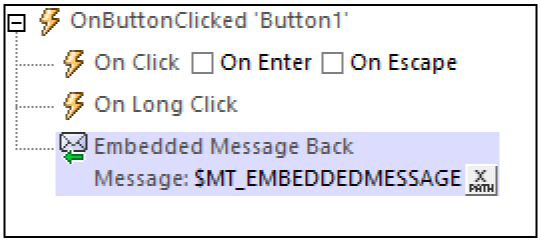
For more information, see the descriptions of **`AddEventListener`** and **`MessageEvent`** at the Mozilla Developer Network website.

### Example

The design contains a `$MT_EMBEDDEDMESSAGE` page source with the structure shown in the screenshot below. Note that the root element of this page source will always be named **`json`** (because this is a JSON page source).



We can send the entire contents of this page source (or a part of it) as a **`message`** event to the solution in the IFrame. This could be done, for example, if a button in the design has an Embedded Message Back [924] action set for its **OnClicked** event (*see screenshot below*). In the XPath expression below, note that it is the contents of **$MT_EMBEDDEDMESSAGE** node (the **json** node and its contents) that is sent as the message event.



OnButtonClicked 'Button1'
On Click □ On Enter □ On Escape
On Long Click
Embedded Message Back
Message: $MT_EMBEDDEDMESSAGE

In the HTML page, we can now register an event listener: `window.addEventListener('message', ProcessReturnMsg)`

We can then access the object as usual and use it in the HTML page. For example:

```
function ProcessReturnMsg(m) {
   msgVar = m.data.json.books
   /* 'm' is the HTML message event that is passed to ProcessReturnMsg */
   /* 'data' belongs to the event and holds the message returned by the MT action */
   /* 'json' is the JSON object that is contained in the message */
   ...
   }
```

In the example above, the content of **books** will be saved to **msgVar**.

# 22.3 Authentication

In order to use a solution, a user needs to be authenticated. Authentication is carried out at two levels: (i) whether a user needs to log in to the server or not; if yes, verify the login credentials; (ii) what kind of permissions are granted to a user when accessing a particular workflow; different users can be assigned different permissions.

Three types of user authentication are available for embedded webpage solutions:

- *Anonymous user* 1438: The user does not need to log in
- *User login* 1439: When the solution loads, the MobileTogether Server login page is displayed in the solution, and the user can log in using credentials that are currently registered with MobileTogether Server
- *JWT authentication* 1439: The authentication is defined outside the MobileTogether system, and is carried out silently without the user having to log in to MobileTogether Server

## Pros and cons of the different authentication methods

The following points should be considered when deciding upon the authentication method for an embedded webpage solution:

- Letting users be anonymous is safe if the solution is used for simple data processing, and does not allow the modification of important databases or the display of sensitive information from databases.
- User login requires the user's login details to be registered with MobileTogether Server and for the user to know the login details.
- User login adds a possibly unwanted layer of interaction between user and solution.
- User login enables users to be authenticated individually.
- JWT authentication is carried out silently, by means of communications that are triggered by code in the webpage. The implementer can decide how to handle the authentication process; this provides flexibility in the design of communication systems.

## One session, one type of authentication

If a session between webpage and server uses one type of authentication, it will continue to use that authentication method till the session is ended or re-started. A session ends when the user logs out or when the server times out (the session timeout is specified in the server settings).

## 22.3.1 Anonymous Login

Anonymous login enables the solution to be displayed in the IFrame without the user having to login or be authenticated in any way. Anonymous login is the simplest authentication method, but should only be set if the solution does not (i) allow the modification of important databases, or (ii) display information that is confidential.

To set anonymous access, do the following:

- Set the server to allow anonymous login (*see the server settings for mobile client ports*)
- Permit anonymous use of the solution (by setting the permissions of the workflow's container to a minimum of `container=read` and `workflow=read,use`).

## 22.3.2 User Login

This type of authentication requires the user to log in to MobileTogether Server with the correct credentials before the solution loads. When the call to the solution is triggered, the MobileTogether Server login page is displayed. For the log in to be authenticated, two criteria need to be fulfilled:

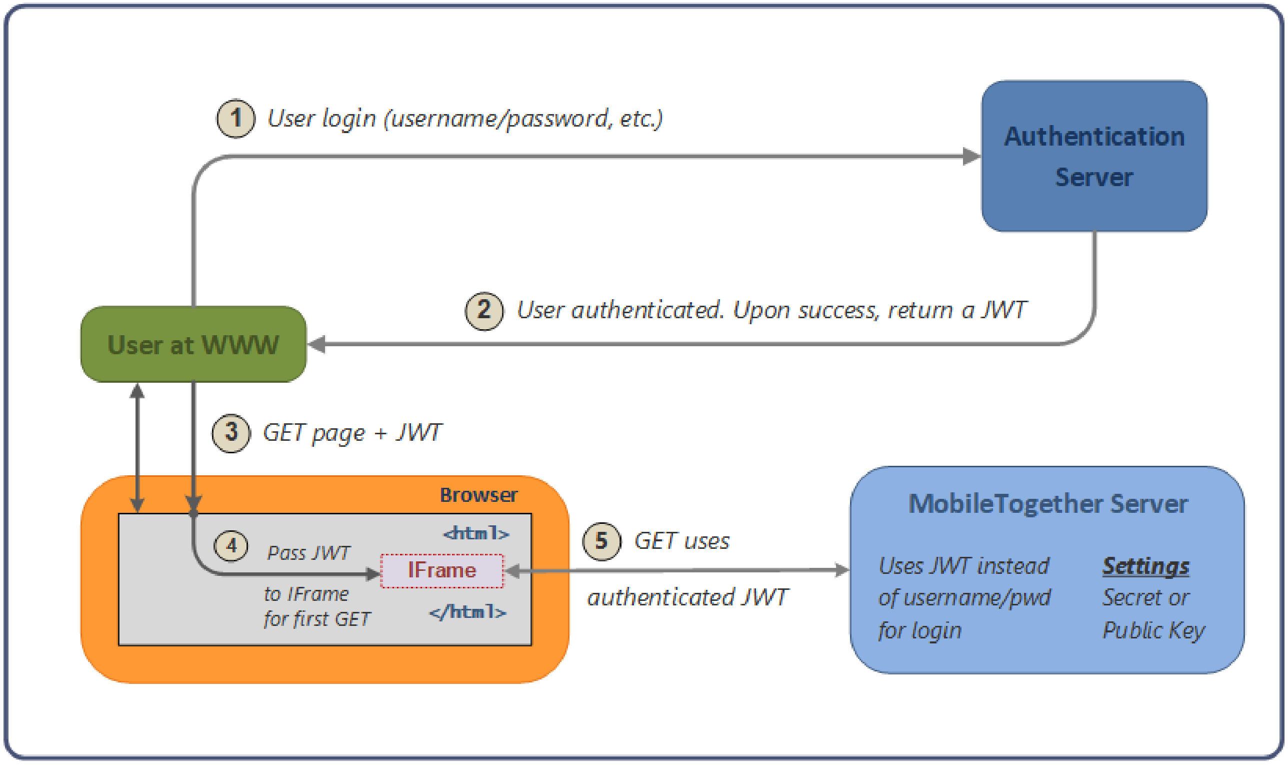
- The user must be a registered user of MobileTogether Server
- The user must have the appropriate permissions to access the targeted solution..

For more information about registering users with MobileTogether Server and setting up their permissions, see the MobileTogether Server documentation.

## 22.3.3 JWT Authentication

An embedded webpage solution can also be authenticated with a JSON Web Tokens (JWT, *recommended pronunciation: "jot"*). Essentially, JWT authentication is processed outside the MobileTogether Server authentication system. An authenticated user is issued with a JWT, which is passed via the webpage to MobileTogether Server. On the server, the JWT is verified; it is also parsed to find out the user. If the JWT is valid, then communication between the embedded solution and the server proceeds for the user specified in the JWT.

The illustration below shows how JWT authentication works.


1 User login (username/password, etc.)
Authentication Server
2 User authenticated. Upon success, return a JWT
User at WWW
3 GET page + JWT
Browser
<html>
4 Pass JWT to IFrame for first GET
IFrame
</html>
5 GET uses authenticated JWT
MobileTogether Server
Uses JWT instead of username/pwd for login
Settings
Secret or Public Key

Note the following points:

- The authentication server and application (www) server do not need to be separate.
- The scenario within which the embedded webpage solution is used, as well as the authentication system that is used, is entirely a matter for the implementer to define.
- When a JWT is created (on successful authentication of a user), a parameter that specifies the user is defined. If the user so specified corresponds to one of the users configured on MobileTogether Server, then server access is determined by the permissions that have been configured for this user. Any other user is automatically imported into the list of configured users; however, permissions for this user will need to be configured on MobileTogether Server.
- The implementer enters the shared secret or public key in the MobileTogether Server settings. The shared secret is the same string of characters that the implementer uses to generate the JWT on the authentication server. The public key is that which corresponds to the private key that is used to encrypt the JWT.
- The JWT is passed to the IFrame, and is passed along with the first call that the solution makes to MobileTogether Server.
- On MobileTogether Server, the shared secret or public key in the MobileTogether Server settings is used to verify the incoming JWT.
- Once the incoming JWT is verified, an authenticated communication session is set up between the solution in the embedded IFrame and MobileTogether Server.
- The user for the session will be that which is specified in the JWT.

## What is a JWT?

A JWT is a set of claims (JSON property–value pairs) that together make up a JSON object. It consists of three parts:

- *Header:* Consists of two properties: `{ "alg": "HS256", "typ": "JWT" }`. `alg` is the algorithm that is used to encrypt the JWT.
- *Payload:* This is where the data to be sent is stored; this data is stored as JSON property–value pairs.
- *Signature:* This is created by encrypting, with the algorithm specified in the header: (i) the base64Url-encoded header, (ii) base64Url-encoded payload, and (iii) a secret (or a private key): `HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret|privateKey)`

The final JWT consists of three parts. Each is base64Url-encoded and separated from the next by a dot. See the `openid.net` and `jwt.io` websites for more details.

### Symmetric key and asymmetric keys

A JWT can be *encrypted* using either a symmetric key (shared secret) or asymmetric keys (the private key of a private–public pair).

- *Symmetric key:* The same key is used for both encryption (when the JWT is created) and decryption (MobileTogether Server uses the key to verify the JWT). The symmetric key—also known as the shared secret—is stored as a setting in MobileTogether Server. *See Symmetric Key: Shared Secret*[1441] *for details of working with symmetric keys*.
- *Asymmetric keys:* Different keys are used for encryption (private key) and decryption (public key). The public key is stored as a setting in MobileTogether Server so that the JWT can be verified. For information about using asymmetric encryption for JWTs, see *Asymmetric Keys: Public Key*[1444].

### Trying out JWT

You could try out JWT as follows:

1. Create your own JWT at the Online JWT Builder of Jamie Kurtz. See the section Symmetric Key: Shared Secret[1441] for a run-through.
2. At the `jwt.io` website, verify your key like this: (i) Enter the encrypted JWT in the *Encoded* pane; (ii) In the *Verify Signature* pane, enter the secret you used to create the JWT. The website's debugger will inform you that the signature has been verified.

Also see the example JWT Authentication[1467].

## 22.3.3.1 Symmetric Key: a Shared Secret

The procedure for using a JWT in an embedded webpage solution:

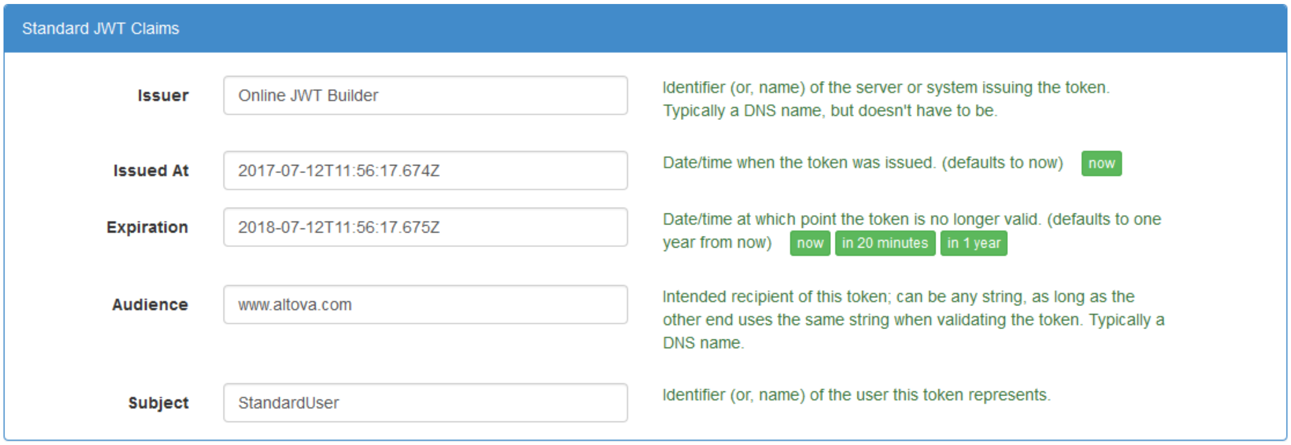
1. Create a JWT[1442] with symmetric encryption. The JWT is based on (i) property–value information that you enter (called claims); and (ii) a random string (the shared secret).
2. Set up MobileTogether Server[1443] to verify the JWT that is sent from the webpage. You provide two pieces of information: (i) the secret that was used to generate the JWT; and (ii) the value of the *Audience* claim (which must be the same as that used to generate the JWT).
3. In the webpage, pass the JWT to the IFrame.

When the JWT is passed to the server, the server validates it by using the *Audience* information and the shared secret that you entered in the settings to generate the JWT.

## Creating a JWT

The description uses the Online JWT Builder of Jamie Kurtz to run through the process of creating a JWT with a symmetric key (shared secret). It describes the claims (JSON property–value pairs) that are relevant for JWT use in the embedded webpage solutions of MobileTogether .

*Standard Claims*
The standard claims *(see screenshot below)* make up the core claims:



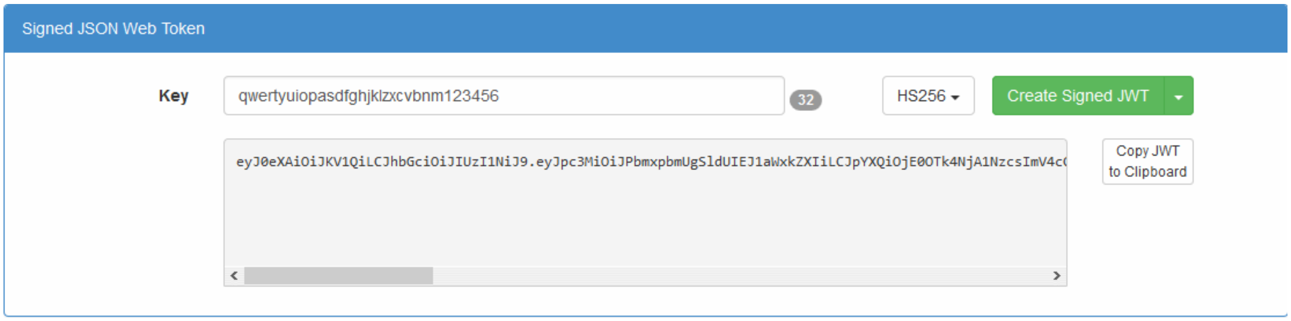
Standard JWT Claims

| | | |
|---|---|---|
| **Issuer** | Online JWT Builder | Identifier (or, name) of the server or system issuing the token. Typically a DNS name, but doesn't have to be. |
| **Issued At** | 2017-07-12T11:56:17.674Z | Date/time when the token was issued. (defaults to now) now |
| **Expiration** | 2018-07-12T11:56:17.675Z | Date/time at which point the token is no longer valid. (defaults to one year from now) now in 20 minutes in 1 year |
| **Audience** | www.altova.com | Intended recipient of this token; can be any string, as long as the other end uses the same string when validating the token. Typically a DNS name. |
| **Subject** | StandardUser | Identifier (or, name) of the user this token represents. |

- MobileTogether Server checks whether the time of server access lies inside the validity period of the JWT. So set the issuance and expiration times as appropriate.
- The *Audience* parameter is one of the settings you need to configure on MobileTogether Server . So specify the same value in both the *Audience* parameter here (when generating the JWT) and the MobileTogether Server *Audience* setting (*see MobileTogether Server Settings* 1443 *below*).
- The *Subject* parameter is where you specify the user that should be logged in to MobileTogether Server. If the user name that you enter here is a user that is registered with MobileTogether Server, then the login is carried out with the permissions that this user has. If the user name is not registered with MobileTogether Server, then this user is registered with MobileTogether Server and logged in; however, you will need to set permissions for this new user so that it can access the relevant workflow.

*Symmetric key (or shared secret) for JWT*
The key (or shared secret), together with the other data you enter, is used to generate the JWT. This secret will be used by MobileTogether Server to decrypt and authenticate the JWT that it receives from the webpage. So the secret is used for both encryption (of the JWT) and its decryption. When generating the JWT, you can specify any string you like as the shared secret. The same string must be entered as the MobileTogether Server *Secret* setting (*see MobileTogether Server Settings* 1443 *below*).

In the screenshot below, a 32-character-long secret is entered, and the encryption algorithm `HS256` is selected. On clicking **Create Signed JWT**, the JWT is created and is displayed in the text box.



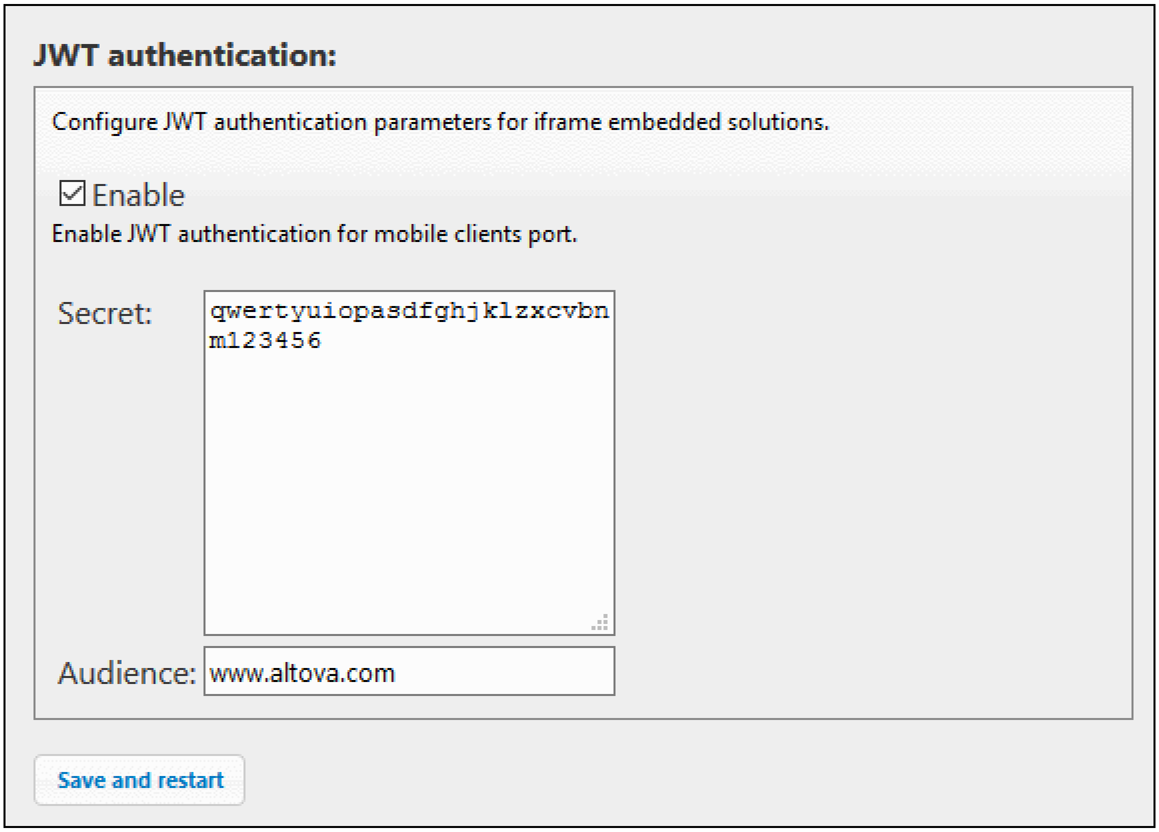
Key: qwertyuiopasdfghjklzxcvbnm123456

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJPbmxpbmUgSldUIEJ1aWxkZXIiLCJpYXQiOjE0OTk4NjA1NzcsImV4c
```

## MobileTogether Server settings

In the *Settings* tab of MobileTogether Server, you will need to enable JWT authentication *(see screenshot below)*, and then enter two settings:

- *Secret:* This is the symmetric key (shared secret) that was used to create the JWT. With this information, the server will be able to verify the JWT. (If you are using asymmetric encryption 1444, then, in this field, enter the public key of a private–public pair.)
- *Audience:* Enter the same string as that you entered for the *Audience* claim when creating the JWT.



**JWT authentication:**

Configure JWT authentication parameters for iframe embedded solutions.

☑ Enable

Enable JWT authentication for mobile clients port.

Secret: qwertyuiopasdfghjklzxcvbnm123456

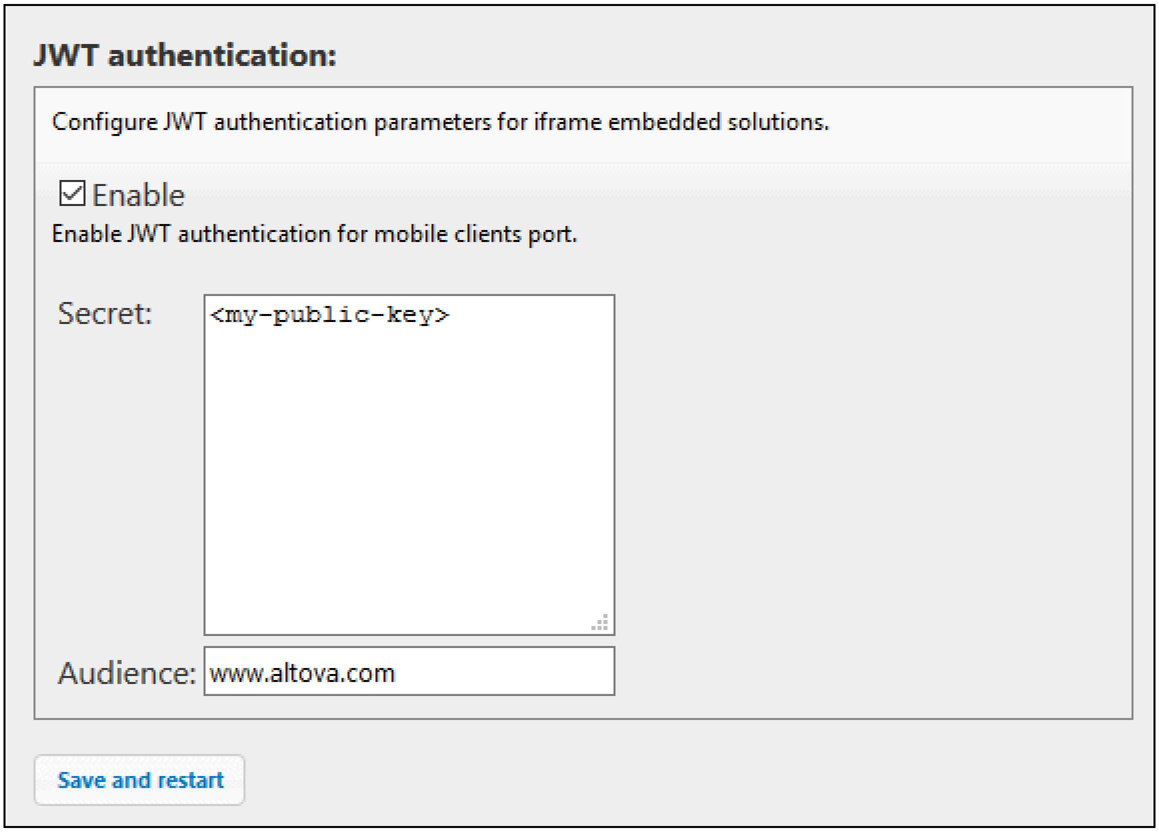
Audience: www.altova.com

### 22.3.3.2 Asymmetric Keys: the Public Key

If you are using asymmetric encryption for your JWT, then the encryption (JWT signing) is done with the private key, and verification is done with the public key. In order for MobileTogether Server to be able to verify the JWT, you must do the following:

In the *Settings* tab of MobileTogether Server, enable JWT authentication *(see screenshot below)*, and then enter settings for:

- *Secret:* Enter the public key of a private–public pair. (If you are using symmetric encryption 1441, enter the shared secret.)
- *Audience:* Enter the same string as that you entered for this claim when creating the JWT.



**JWT authentication:**

Configure JWT authentication parameters for iframe embedded solutions.

☑ Enable

Enable JWT authentication for mobile clients port.

Secret: `<my-public-key>`

Audience: www.altova.com

# 22.4 Examples

The examples in this section show how to set up an embedded webpage solution, starting with a simple example and ending with an example that uses JWT authentication. The files described in this section are available in your (72 My) Documents 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions`**.

In this section:

- Embedding a Solution 1445
- Sending/Receiving JSON Data 1445
- Sending/Receiving XML Data 1453
- Pre-setting the JSON Page Source 1461
- JWT Authentication 1467

## 22.4.1 Embedding a Solution

The HTML code listed below shows how an **`IFrame`** element is used to embed a solution. You can try out this feature by copying the HTML code below and saving it to file, and then opening the file in a browser.

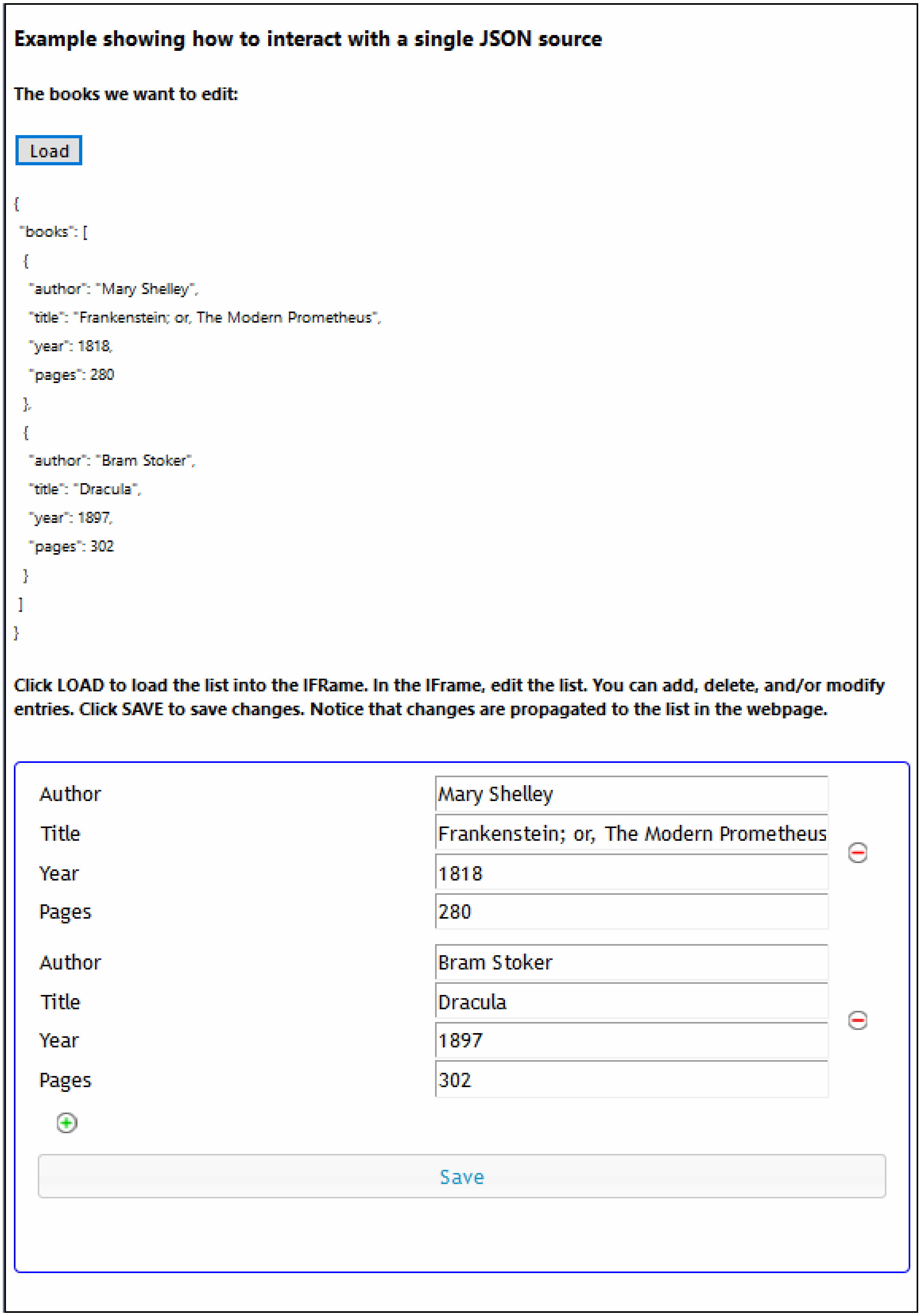
```
<!DOCTYPE html>
<html>
  <head>
    <title>Webpage containing an embedded solution</title>
  </head>
  <body>
    <h1>Webpage containing an embedded solution</h1>
    <p>The embedded solution is loaded into an IFrame, which is located immediately below
this paragraph.</p>
    <div class="resize">
      <iframe src="http://localhost:8083/run?d=/public/About" frameborder="0"></iframe>
    </div>
  </body>
</html>
```

The solution used in this listing is a sample named `About` that is packaged with MobileTogether Server; it is located by default in the server's `public` container. In order to correctly embed this solution, set up the server to allow anonymous access 1438 of the `About` workflow.

*See also: Embedding a Solution in a Webpage* 1431

## 22.4.2 Sending/Receiving JSON Data

This section explains the working of an embedded webpage solution that uses JSON source data. A list of books in JSON format is sent from the webpage (*screenshot below*) to an embedded IFrame (*framed in blue*). Here, the list can be edited using a MobileTogether solution. On saving the changes in the IFrame, the edited book list is sent to the webpage.



**Example showing how to interact with a single JSON source**

**The books we want to edit:**

```
{
 "books": [
  {
   "author": "Mary Shelley",
   "title": "Frankenstein; or, The Modern Prometheus",
   "year": 1818,
   "pages": 280
  },
  {
   "author": "Bram Stoker",
   "title": "Dracula",
   "year": 1897,
   "pages": 302
  }
 ]
}
```

**Click LOAD to load the list into the IFRame. In the IFrame, edit the list. You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that changes are propagated to the list in the webpage.**

| | |
|---|---|
| Author | Mary Shelley |
| Title | Frankenstein; or, The Modern Prometheus |
| Year | 1818 |
| Pages | 280 |
| Author | Bram Stoker |
| Title | Dracula |
| Year | 1897 |
| Pages | 302 |

The embedded webpage solution consists of the HTML webpage (`jsonBooks.html`) and a MobileTogether design (`jsonBooks.mtd`). Both files are located in your (72 My) Documents 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions`**. To try out the files, deploy the MTD file to your server and enable it to be accessed anonymously 1438. If needed, modify the HTML code so that the IFrame correctly targets the workflow on the server 1445. Open the webpage in a browser and click the **`Load`** button to start.

The description below contains the complete HTML code listing of the webpage, followed by a color-coded explanation of how the HTML code interacts with the solution.

## HTML code listing

The HTML code listing of the file `jsonBooks.html`. An explanation of the code is given in the next section below. Please note that some JavaScript functionality used in this example might not be available in all browsers. In this case, please modify the JavaScript to suit your browser.

*Webpage code listing*

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         * {
            font-family: Segoe UI, Tahoma, Arial, Helvetica, sans-serif;
         }
         iframe {
            width: 100%;
            height: 400px;
            border: 2px solid blue;
            border-radius: 5px;
            margin: 10px 0px;
         }
         code {
            font-size: small;
         }
      </style>
      <script>
         // The initial book list is stored in a variable in JSON format
         // It can be easily handled in JavaScript
            var books = {
               "books": [
                  {
                     "author": "Mary Shelley",
                     "title": "Frankenstein; or, The Modern Prometheus",
                     "year": 1818,
                     "pages": 280
                  },
                  {
                     "author": "Bram Stoker",
                     "title": "Dracula",
                     "year": 1897,
                     "pages": 302
                  }
               ]
            };

            // Posts variable 'books' to IFrame (from where 'books' is forwarded to MT
Server)
            function sendbooks()
{
               document.querySelector('iframe').contentWindow.postMessage(books,
'*');
            }


           // Contents of variable 'books' converted to string and displayed inside HTML
element CODE
            function showbooks() {
               document.querySelector('code').innerText = JSON.stringify(books, null, '
');
            }

            // m = HTML message event; data = container for message from server
            // m.data.json = contents of the 'json' object that was sent from the server
            function receivebooks(m) {
               books = m.data.json;
               showbooks();
            }

            // Handler to receive messages from server via solution in IFrame
            window.addEventListener('message', receivebooks);

            // Handler to show initial book list in webpage on page load
            document.addEventListener('DOMContentLoaded', showbooks);

      </script>
   </head>


   <body>
      <h4>Example showing how to interact with a single JSON source</h4>
      <h5>The books we want to edit:</h5>
      <!-- Send the JSON book list from the webpage to the IFrame -->
      <button onclick="sendbooks()">Load</button>
      <pre><code><!-- The SHOWBOOKS function displays the book list here --></code></pre>
      <h5>
         Click LOAD to load the book list into the IFrame. In the IFrame, edit the list.
You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that
changes are propagated to the list in the webpage.
      </h5>
      <iframe src="http://localhost:8083/run?d=/public/jsonBooks"
frameborder="0"></iframe>
   </body>

</html>
```

## How it works

In this explanatory part, different background colors are used to indicate what's happening in the individual parts of the mechanism (*webpage–solution–workflow*):

| | |
|---|---|
| 🟫 | ***Webpage:** user actions and how the HTML/JavaScript code works* |
| 🟦 | ***Solution:** actions carried out by the solution in the IFrame* |
| 🟩 | ***Workflow:** processing on the server (based on the MT design)* |

*On loading the HTML page*:

A JavaScript variable named **books** is read. It contains a JSON object named **books**.

```
var books = {
   "books": [
      {
         "author": "Mary Shelley",
         "title": "Frankenstein; or, The Modern Prometheus",
         "year": 1818,
         "pages": 280
      },
      {
         "author": "Bram Stoker",
         "title": "Dracula",
         "year": 1897,
         "pages": 302
      }
   ]
};
```

The content of the **books** variable is displayed inside the HTML **code** element by using an event listener (that listens for a **DOMContentLoaded** event) and a JavaScript function (**showbooks**):

```
document.addEventListener('DOMContentLoaded', showbooks);

function showbooks() {
   document.querySelector('code').innerText = JSON.stringify(books, null, ' ');
}

<pre><code></code></pre>
```

This enables us to see the contents of the **books** variable when the HTML document is loaded. (Later, we will use the same JavaScript function to check whether the **books** variable has been updated.)

The IFrame loads the solution **jsonBooks** (targeted in the **src** attribute of the **iframe** element):

```
<iframe src="http://localhost:8083/run?d=/public/jsonBooks" frameborder="0">
</iframe>
```

*On clicking the **Load** button:*

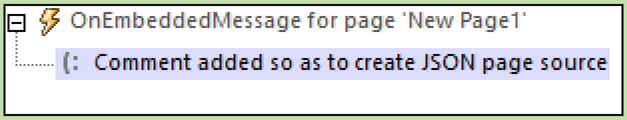
```
<button onclick="sendbooks()">Load</button>
```

A JavaScript function uses **postMessage()** to send the contents of the **books** variable to the IFrame.

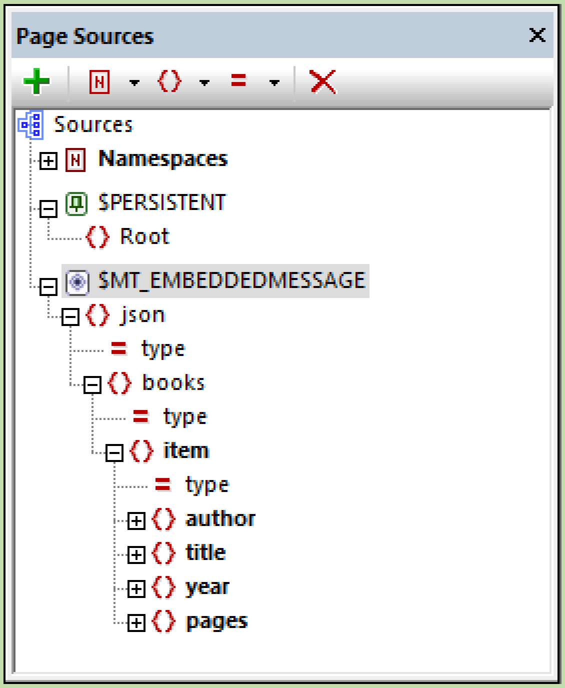
```
function sendbooks() {
   document.querySelector('iframe').contentWindow.postMessage(books, '*');
   }
```

**{books}** is automatically sent to the workflow on the server (in serialized JSON form).

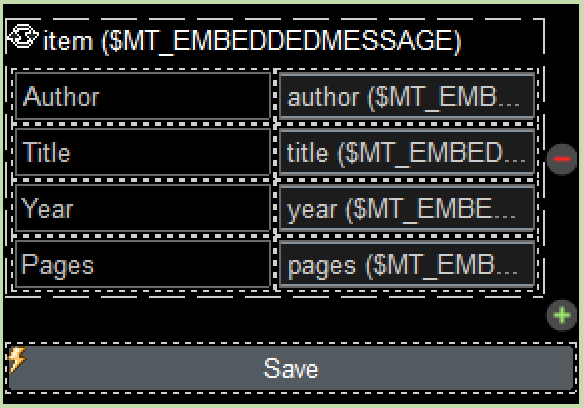
Since page event **OnEmbeddedMessage** has an action defined, ...


OnEmbeddedMessage for page 'New Page1'
(: Comment added so as to create JSON page source

... the page source **$MT_EMBEDDEDMESSAGE** is automatically loaded with the **{books}** data. For this to work as intended in the design, the structure of the page source (defined at design time) must correspond to the structure of the incoming JSON data. Note that the **item** node in the page source corresponds to each item in the JSON array. (If the incoming JSON data does not match the structure defined for the page source, it will be loaded anyway—with its own structure. But since XPath expressions in the design will reference the defined structure, the structure loaded at runtime will not be reached by these XPath expressions.)

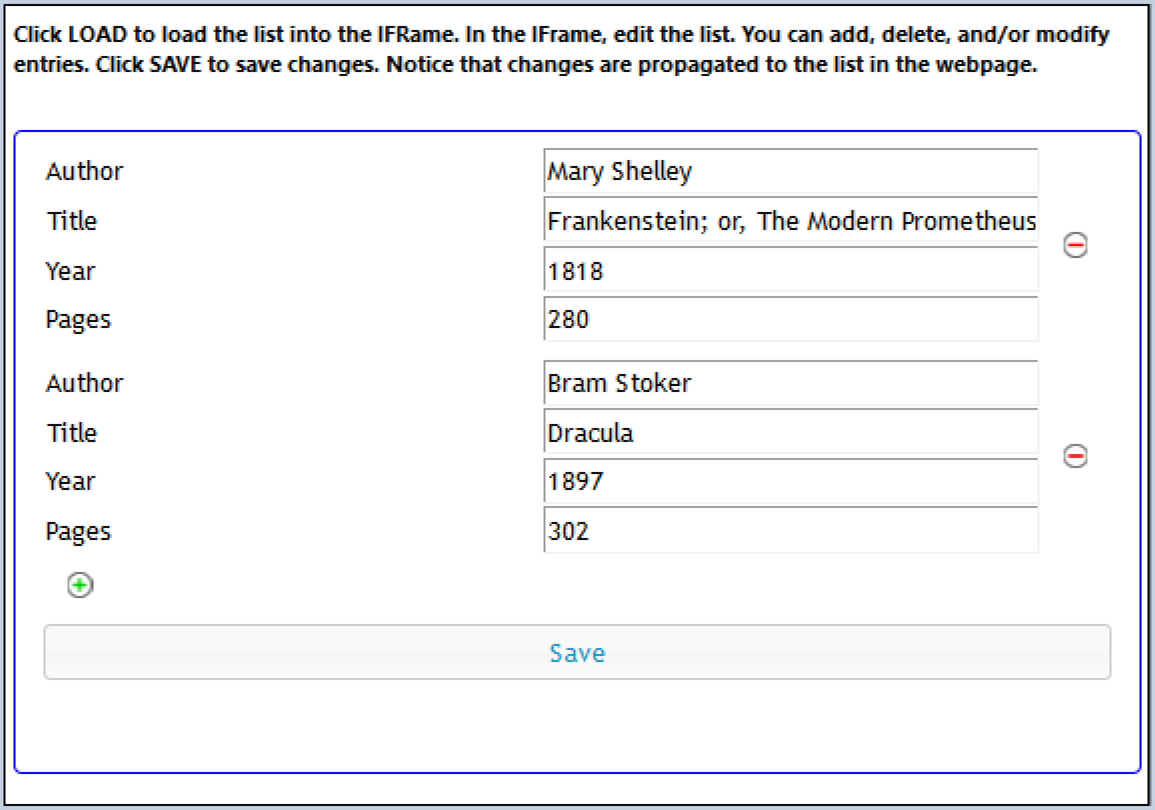


The design contains a repeating table of **item** nodes. Cells of the table are linked, respectively, to the **author**, **title**, **year**, and **pages** page source nodes.



As a result,...

...the solution in the IFrame is updated. The repeating table is populated with the data in the workflow's **$MT_EMBEDDEDMESSAGE** page source.

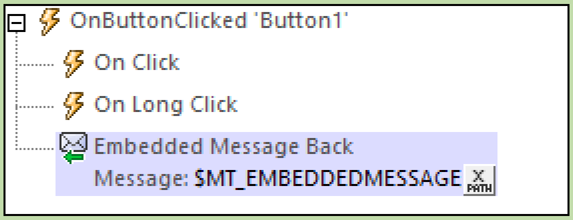


Click LOAD to load the list into the IFRame. In the IFrame, edit the list. You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that changes are propagated to the list in the webpage.

| | |
|---|---|
| Author | Mary Shelley |
| Title | Frankenstein; or, The Modern Prometheus |
| Year | 1818 |
| Pages | 280 |
| Author | Bram Stoker |
| Title | Dracula |
| Year | 1897 |
| Pages | 302 |

On editing the books data in the solution, the **$MT_EMBEDDEDMESSAGE** page source is continually updated. On clicking **Save**, an **OnButtonClick** event handler is triggered.

The **OnButtonClick** event specifies that an Embedded Message Back action be performed. This sends the contents of the entire page source **$MT_EMBEDDEDMESSAGE** as a **message** event to the IFrame. (Note that **$MT_EMBEDDEDMESSAGE** now contains the edited book list.)



OnButtonClicked 'Button1'
On Click
On Long Click
Embedded Message Back
Message: $MT_EMBEDDEDMESSAGE

An event listener has been registered to listen for the **message** event. On picking up a **message** event, a JavaScript function (**receivebooks**) is called:

```
window.addEventListener('message', receivebooks);
```

The **receivebooks** function (*see below*) takes the **message** event **(m)** as its parameter (**data** is the data of the **message** event), and assigns the content of the **json** object in the received message to the

**books** variable. The **books** variable now contains the contents of the **json** object, which is the updated book list from the server. The structure of the modified book list is the same as that of the original book list (but with more or fewer book items in the **books** array).

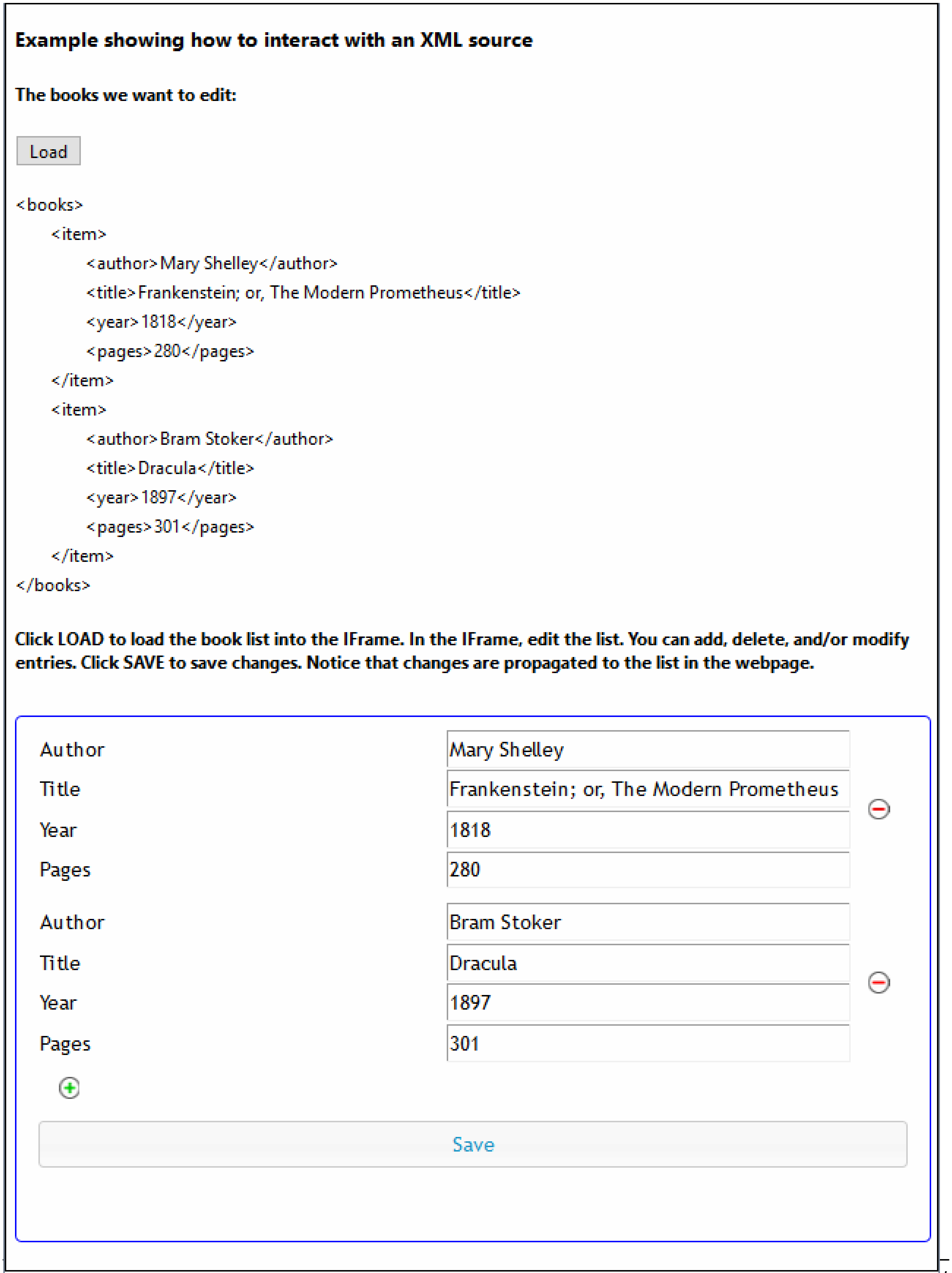
```
function receivebooks(m) {
   books = m.data.json;
   showbooks();
}
```

The **showbooks** function displays the updated book list in the webpage:

```
function showbooks() {
   document.querySelector('code').innerText = JSON.stringify(books, null, ' ');
}

<pre><code></code></pre>
```

## 22.4.3 Sending/Receiving XML Data

This section explains the working of an embedded webpage solution that uses XML source data (a book list). The book list is sent from the webpage (*screenshot below*) to an embedded IFrame (*framed in blue*). Here, the list can be edited using a MobileTogether solution. On saving the changes in the IFrame, the edited book list is sent to the webpage.



Example showing how to interact with an XML source

The books we want to edit:

```
<books>
    <item>
        <author>Mary Shelley</author>
        <title>Frankenstein; or, The Modern Prometheus</title>
        <year>1818</year>
        <pages>280</pages>
    </item>
    <item>
        <author>Bram Stoker</author>
        <title>Dracula</title>
        <year>1897</year>
        <pages>301</pages>
    </item>
</books>
```

Click LOAD to load the book list into the IFrame. In the IFrame, edit the list. You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that changes are propagated to the list in the webpage.

| | |
|---|---|
| Author | Mary Shelley |
| Title | Frankenstein; or, The Modern Prometheus |
| Year | 1818 |
| Pages | 280 |
| Author | Bram Stoker |
| Title | Dracula |
| Year | 1897 |
| Pages | 301 |

The embedded webpage solution consists of the HTML webpage (`xmlBooks.html`) and a MobileTogether design (`xmlBooks.mtd`). Both files are located in your ([72]My) Documents[72] MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions`**. To try out the files, deploy the MTD file to your server and enable it to be accessed anonymously[1438]. If needed, modify the HTML code so that the IFrame correctly targets the workflow on the server[1445]. Open the webpage in a browser and click the **Load** button to start.

The description below contains the complete HTML code listing of the webpage, followed by a color-coded explanation of how the HTML code interacts with the solution.

## HTML code listing

The HTML code listing of the file `xmlBooks.html`. An explanation of the code is given in the next section below. Please note that some JavaScript functionality used in this example might not be available in all browsers. In this case, please modify the JavaScript to suit your browser.

*Webpage code listing*

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         * {
            font-family: Segoe UI, Tahoma, Arial, Helvetica, sans-serif;
         }
         iframe {
            width: 100%;
            height: 400px;
            border: 2px solid blue;
            border-radius: 5px;
            margin: 10px 0px;
         }
         code {
            font-size: small;
         }
      </style>
      <script>
         // The book list in XML format
         var books = '
         <books>
            <item>
               <author>Mary Shelley</author>
               <title>Frankenstein; or, The Modern Prometheus</title>
               <year>1818</year>
               <pages>280</pages>
            </item>
            <item>
               <author>Bram Stoker</author>
               <title>Dracula</title>
               <year>1897</year>
               <pages>302</pages>
            </item>
         </books>
         ';
         // This is the XML DOM tree (initialized in showbooks)
         var books;

         function sendbooks() {
            document.querySelector('iframe').contentWindow.postMessage({
            "books": books.childNodes[0].outerHTML
            }, '*');
         }

         // This is the function that receives the updated books
         function receivebooks(m) {
            books = m.data.json.books;
            showbooks();
         }

         // Contents of variable 'books' converted to string and displayed inside HTML
element CODE
         function showbooks() {
            // Create a DOM tree from the XML
            books = new DOMParser().parseFromString(books, 'text/xml');
            // Manipulate the DOM and show the result
            document.querySelector('code').innerText = books.childNodes[0].outerHTML;
         }

         // Handler to receive messages from server via solution in IFrame
         window.addEventListener('message', receivebooks);

         // Handler to show initial list of books on page load
         document.addEventListener('DOMContentLoaded', showbooks);
     </script>
   </head>
   <body>
      <h4>Example showing how to interact with an XML source</h4>
      <h5>The books we want to edit:</h5>
      <button onclick="sendbooks()">Load</button>
      <pre><code></code></pre>
      <h5>
         Click LOAD to load the book list into the IFrame. In the IFrame, edit the list.
You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that
changes are propagated to the list in the webpage.
      </h5>
      <iframe src="http://localhost:8083/run?d=/public/xmlBooks"
frameborder="0"></iframe>
   </body>
</html>
```

## How it works

In this explanatory part, different background colors are used to indicate what's happening in the individual parts of the mechanism (*webpage–solution–workflow*):

| | |
|---|---|
| 🟫 | ***Webpage:*** *user actions and how the HTML/JavaScript code works* |
| 🟦 | ***Solution:*** *actions carried out by the solution in the IFrame* |
| 🟩 | ***Workflow:*** *processing on the server (based on the MT design)* |

*On loading the HTML page*:

A JavaScript variable named **books** is read. It contains a string containing an XML structure:

```
var books = '
   <books>
      <item>
         <author>Mary Shelley</author>
         <title>Frankenstein; or, The Modern Prometheus</title>
         <year>1818</year>
         <pages>280</pages>
      </item>
      <item>
         <author>Bram Stoker</author>
         <title>Dracula</title>
         <year>1897</year>
         <pages>302</pages>
      </item>
   </books>
';
```

The content of the **books** variable is displayed inside the HTML **code** element by using an event listener (that listens for a **DOMContentLoaded** event) and a JavaScript function (**showbooks**):

```
document.addEventListener('DOMContentLoaded', showbooks);
```

The **showbooks** function: (i) creates a DOM tree from the XML structure in the **books** variable, and (ii) places the desired XML structure inside the HTML **code** element.

```
function showbooks() {
   books = new DOMParser().parseFromString(books, 'text/xml');
   document.querySelector('code').innerText = books.childNodes[0].outerHTML;
}
```

```
<pre><code></code></pre>
```

This enables us to see the contents of the **books** variable when the HTML document is loaded. (Later, we will use the **showbooks** function to check whether the **books** variable has been updated.)

The IFrame loads the solution **xmlBooks** (targeted in the **src** attribute of the **iframe** element):

```
<iframe src="http://localhost:8083/run?d=/public/xmlBooks" frameborder="0"></iframe>
```

*On clicking the **Load** button:*

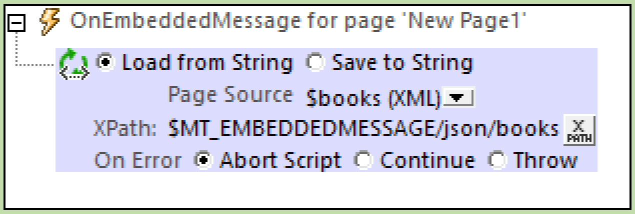
```
<button onclick="sendbooks()">Load</button>
```

A JavaScript function (**sendbooks**) uses **postMessage()** to send the contents of the **books** variable to the IFrame. Note that the XML content is placed inside a JSON object. (This is because the workflow expects to receive JSON.)

```
function sendbooks() {
   document.querySelector('iframe').contentWindow.postMessage({
      "books": books.childNodes[0].outerHTML
   }, '*');
}
```

**{books}** is automatically sent to the workflow on the server (in serialized JSON form).

The page event **OnEmbeddedMessage** is enabled since an action has been defined for it (*see screenshot below*). As a result, the page source **$MT_EMBEDDEDMESSAGE** is automatically loaded with **{books}** data.



OnEmbeddedMessage for page 'New Page1'

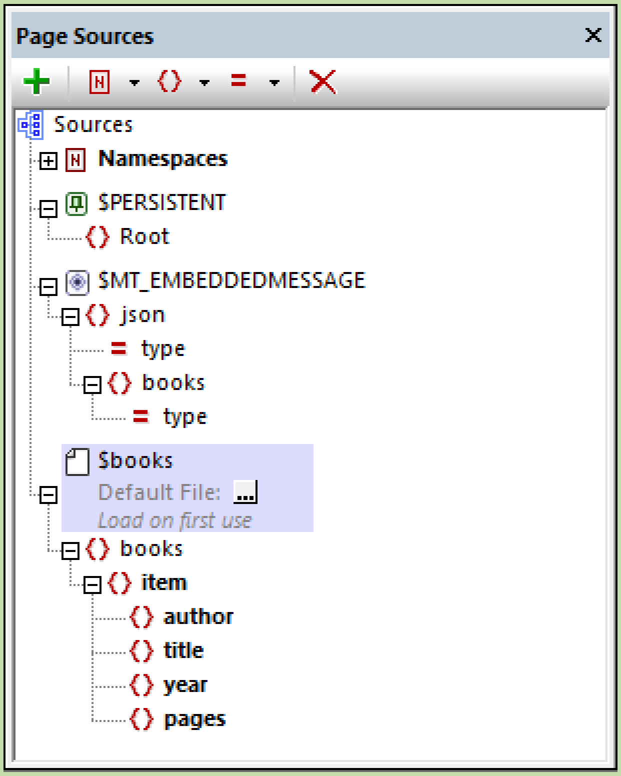
● Load from String ○ Save to String

Page Source $books (XML)

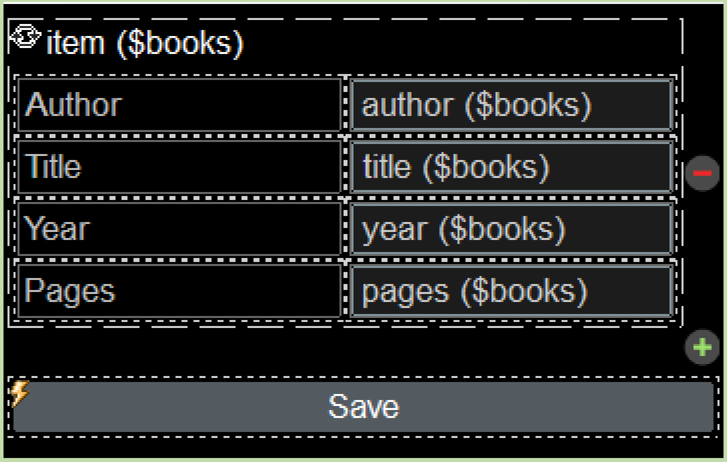
XPath: $MT_EMBEDDEDMESSAGE/json/books

On Error ● Abort Script ○ Continue ○ Throw

The Load from String action (*screenshot above*) creates the content of **$MT_EMBEDDEDMESSAGE/json/books** as the XML page source **$books**. The structures of both these page sources have been created beforehand (*see screenshot below*).

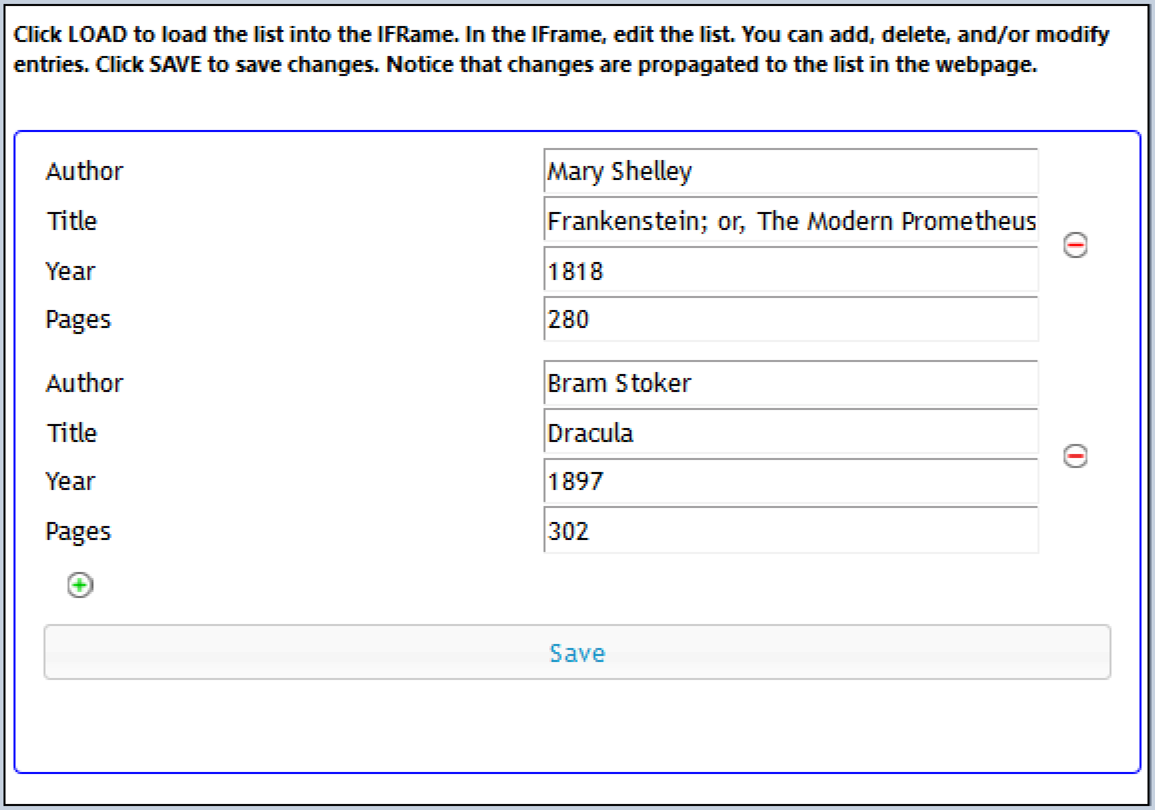


The XML page source has been created so that the XML data can be linked to design components, thereby enabling the XML data to be edited. The design contains a repeating table of **`item`** nodes of the **`$books`** page source (*see screenshot below*). Cells of the table are linked, respectively, to the **`author`**, **`title`**, **`year`**, and **`pages`** page source nodes.



Because the page source has been loaded, the repeating table is populated with the data in the workflow's **`$MT_EMBEDDEDMESSAGE`** page source.

This data update is shown in the solution in the IFrame.

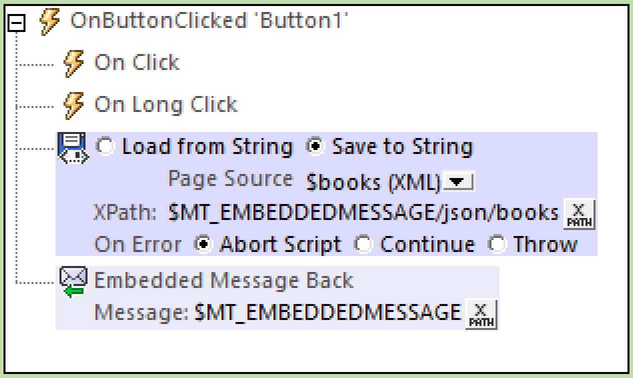


Click LOAD to load the list into the IFRame. In the IFrame, edit the list. You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that changes are propagated to the list in the webpage.

| | |
|---|---|
| Author | Mary Shelley |
| Title | Frankenstein; or, The Modern Prometheus |
| Year | 1818 |
| Pages | 280 |
| Author | Bram Stoker |
| Title | Dracula |
| Year | 1897 |
| Pages | 302 |

On editing the books data in the solution, the **$books** page source is continually updated. On clicking **Save**, an **OnButtonClick** event handler is triggered.

The **OnButtonClick** event specifies two actions: (i) a Save to String action, which saves the **$books** page source (containing the edited book list) to **$MT_EMBEDDEDMESSAGE/json/books**; (ii) an Embedded Message Back action, which sends the contents of **$MT_EMBEDDEDMESSAGE** as a **message** event to the IFrame. (Note that **$MT_EMBEDDEDMESSAGE/json/books** contains the edited book list.)



OnButtonClicked 'Button1'
- On Click
- On Long Click
- ○ Load from String ● Save to String
  Page Source $books (XML)
  XPath: $MT_EMBEDDEDMESSAGE/json/books
  On Error ● Abort Script ○ Continue ○ Throw
- Embedded Message Back
  Message: $MT_EMBEDDEDMESSAGE

An event listener has been registered to listen for the **message** event. On picking up a **message** event, a JavaScript function (**receivebooks**) is called:

```
window.addEventListener('message', receivebooks);
```

The **receivebooks** function (*see below*) takes the **message** event **(m)** as its parameter (**data** is the data of the **message** event), and assigns the content of the **json/books** object in the received message to the **books** variable. The **books** variable now contains the updated book list from the server.

```
function receivebooks(m) {
   books = m.data.json.books;
   showbooks();
}
```

The **showbooks** function: (i) creates a DOM tree from the XML structure in the **books** variable, and (ii) places the desired XML structure inside the HTML **code** element.

```
function showbooks() {
   books = new DOMParser().parseFromString(books, 'text/xml');
   document.querySelector('code').innerText = books.childNodes[0].outerHTML;
}
```

```
<pre><code></code></pre>
```

The updated book list is shown in the webpage.

## 22.4.4 Pre-setting the JSON Page Source

The example in this section shows how to automatically send JSON source data from the webpage to the workflow when the HTML page is opened.

The HTML code is based on that used in the Sending/Receiving JSON Data example[1445]. The difference is this: In the former example, the user must click a button in the webpage to send the initial book list to the IFrame; in this example, the data is automatically loaded when the page is opened. (The **Load** button and its function have been removed, and a new function is used to automatically load the data.)

The files used in this example are `jsonBooksOnStart.html` and `jsonBooks.mtd`. Both are located in your ([72] *My) Documents*[72] MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions**. To try out the files, deploy the MTD file to your server and enable it to be accessed anonymously[1438]. If needed, modify the HTML code so that the correct workflow is targeted.

The description below explains only those points that relate to the automatic loading of the JSON data. For an explanation of the other aspects of the mechanism, see Sending/Receiving JSON Data[1445].

## HTML code listing

The HTML code listing of the file `jsonBooksOnStart.html`. An explanation of the code is given in the next section below. Please note that some JavaScript functionality used in this example might not be available in all browsers. In this case, please modify the JavaScript to suit your browser.

*Webpage code listing*

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         * {
            font-family: Segoe UI, Tahoma, Arial, Helvetica, sans-serif;
         }
         iframe {
            width: 100%;
            height: 400px;
            border: 2px solid blue;
            border-radius: 5px;
            margin: 10px 0px;
         }
         code {
            font-size: small;
         }
      </style>
      <script src="http://localhost:8083/js/WebAppIFrame.js"></script>
      <script>
         // The initial book list stored in a variable in JSON format
            var books = {
               "books": [
                  {
                     "author": "Mary Shelley",
                     "title": "Frankenstein; or, The Modern Prometheus",
                     "year": 1818,
                     "pages": 280
                  },
                  {
                     "author": "Bram Stoker",
                     "title": "Dracula",
                     "year": 1897,
                     "pages": 302
                  }
               ]
            };

            // Contents of variable 'books' converted to string and displayed inside HTML
element CODE
            function showbooks() {
               document.querySelector('code').innerText = JSON.stringify(books, null, '
');
            }
            // m = HTML message event; data = container for message from server
            // m.data.json = contents of the 'json' object that was sent from the server
            function receivebooks(m) {
               books = m.data.json;
               showbooks();
            }

            // Handler to show books in webpage on page load
            document.addEventListener('DOMContentLoaded', showbooks);

            // Handler to receive messages from server via solution in IFrame
            window.addEventListener('message', receivebooks);

            // Handler to send data to IFrame on page load
            document.addEventListener('DOMContentLoaded', function() {
               var embedded = new WebAppIFrame(document.querySelector('iframe'));

               embedded.start('http://localhost:8083/run?d=/public/jsonBooks', books);
            });
      </script>
   </head>


   <body>
      <h4>An editable list of books in JSON format</h4>
      <h5>The book list, stored as a JSON object in the webpage:</h5>
      <pre><code><!-- The SHOWBOOKS function displays the book list here --></code></pre>
      <h5>The book list is displayed in the Iframe as soon as the HTML page is
opened.</h5>
      <iframe frameborder="0"></iframe>
   </body>


</html>
```

## How it works

In this part, different background colors are used to indicate what's happening in the individual parts of the mechanism (*webpage–solution–workflow*):

| | |
|---|---|
| 🟫 | ***Webpage:*** *user actions and how the HTML/JavaScript code works* |
| 🟦 | ***Solution:*** *actions carried out by the solution in the IFrame* |
| 🟩 | ***Workflow:*** *processing on the server (based on the MT design)* |

*On loading the HTML webpage*:

A JavaScript variable named **books** is read. It contains a JSON object named **books**.

```
var books = {
   "books": [
      {
         "author": "Mary Shelley",
         "title": "Frankenstein; or, The Modern Prometheus",
         "year": 1818,
         "pages": 280
      },
      {
         "author": "Bram Stoker",
         "title": "Dracula",
         "year": 1897,
         "pages": 302
      }
   ]
};
```

*Displaying the book list in the webpage*:

The content of the **books** variable is displayed inside the HTML **code** element by using an event listener (that listens for a **DOMContentLoaded** event) and a JavaScript function (**showbooks**):

```
document.addEventListener('DOMContentLoaded', showbooks);

function showbooks() {
   document.querySelector('code').innerText = JSON.stringify(books, null, ' ');
}

<pre><code></code></pre>
```

This enables us to see the book list in the webpage when the HTML document is loaded.

*Automatically sending the book list to the IFrame on loading the webpage*:

An event listener that listens for a **DOMContentLoaded** event defines the automatic-load function:

```
document.addEventListener('DOMContentLoaded', function() {
   var embedded = new WebAppIFrame(document.querySelector('iframe'));
   embedded.start('http://localhost:8083/run?d=/public/jsonBooks', books);
});
```

The function defined above creates a variable by calling **WebAppIFrame.js**. Notice that the reference to the JavaScript file (see **script** element below) was not required in the previous JSON example[1445].

```
<script src="http://localhost:8083/js/WebAppIFrame.js"></script>
```

**WebAppIFrame.js** *(listing below)* contains code to simplify loading the solution and sending the data to **$MT_EMBEDDEDMESSAGE**. Notice that the URL to start the solution is not given in the **src** attribute of the IFrame, but is passed as the first parameter of the **start** method..

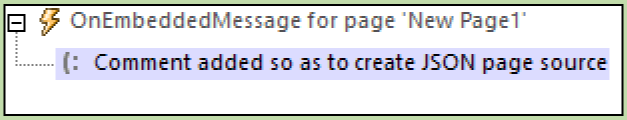
*Code listing of WebAppIFrame.js:*

```
'use strict';

function WebAppIFrame(iframe, listener) {
   var _this = this;
   var _data;
   var _jwt;

   if (listener) {
      window.addEventListener('message', listener, false);
   }

   this.start = function(url, data, jwt) {
      function _start() {
         _this.post({data: _data, jwt: _jwt});
            iframe.removeEventListener('load', _start);
      }

      _data = data;
      _jwt = jwt;
   if (_jwt) {
         url += '&auth';
      }
         iframe.addEventListener('load', _start);
            iframe.src = url + '&embed';
   }

   this.post = function(data) {
      iframe.contentWindow.postMessage(data, '*');
   }
}
```

The IFrame loads the solution **jsonBooks** and receives the data from the webpage.

**{books}** is automatically sent to the workflow on the server (in serialized JSON form).

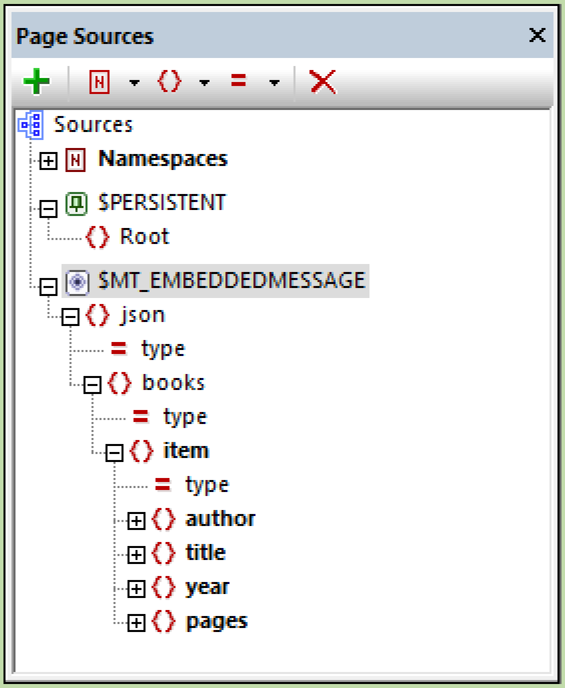
Since page event **OnEmbeddedMessage** has an action defined, ...



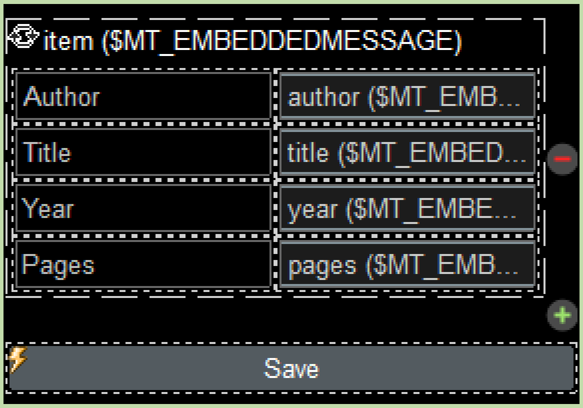
OnEmbeddedMessage for page 'New Page1'

(: Comment added so as to create JSON page source

... the page source **$MT_EMBEDDEDMESSAGE** is automatically loaded with **{books}** data.

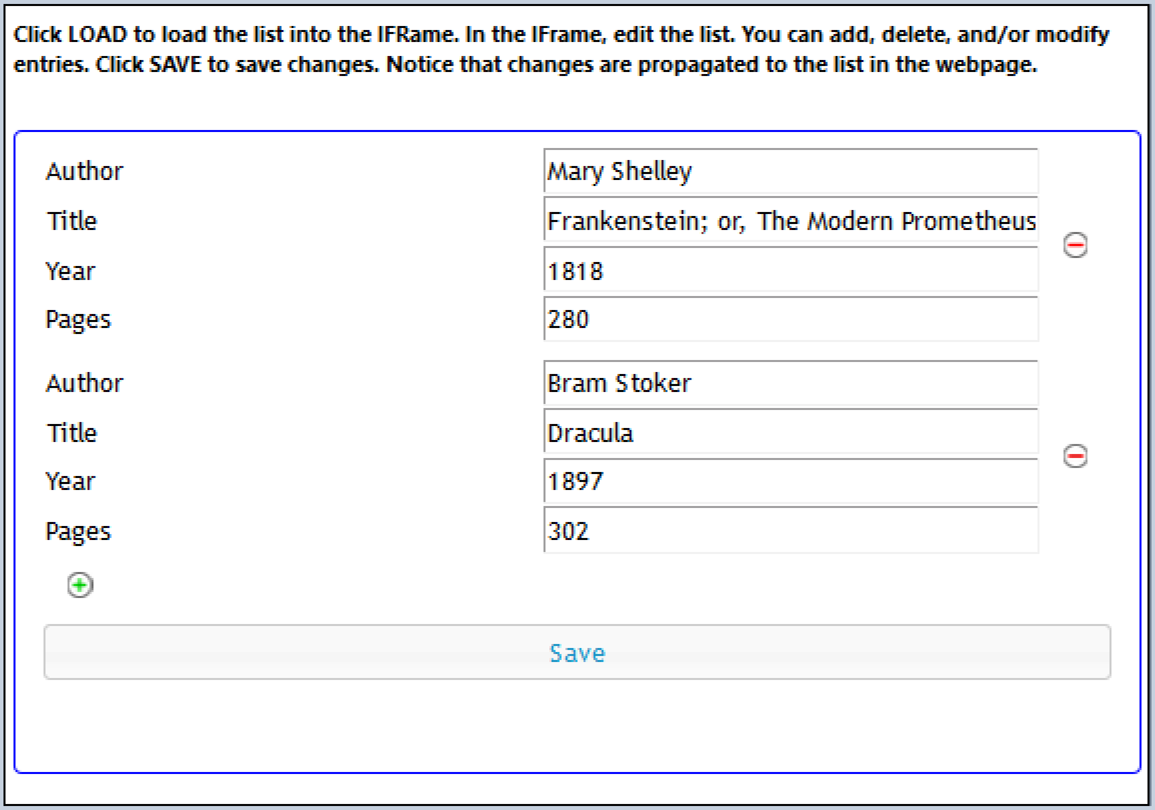


The design contains a repeating table of **`item`** nodes. Cells of the table are linked, respectively, to the **`author`**, **`title`**, **`year`**, and **`pages`** page source nodes.



As a result,...

...the solution in the IFrame is loaded with this data. The repeating table is populated with the data in the workflow's **`$MT_EMBEDDEDMESSAGE`** page source.



**Click LOAD to load the list into the IFRame. In the IFrame, edit the list. You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that changes are propagated to the list in the webpage.**

| | |
|---|---|
| Author | Mary Shelley |
| Title | Frankenstein; or, The Modern Prometheus |
| Year | 1818 |
| Pages | 280 |
| Author | Bram Stoker |
| Title | Dracula |
| Year | 1897 |
| Pages | 302 |

The data has been made available in the IFrame directly on opening the HTML page.

## 22.4.5 JWT Authentication

The JWT authentication example in this section modifies the webpage of the Pre-setting page source example in the previous section 1461. Together with the call to the solution, we also submit the JWT. Note that the JWT must be submitted as a string (that is, with quotes around it). In the code listing below, the JWT is highlighted in blue.

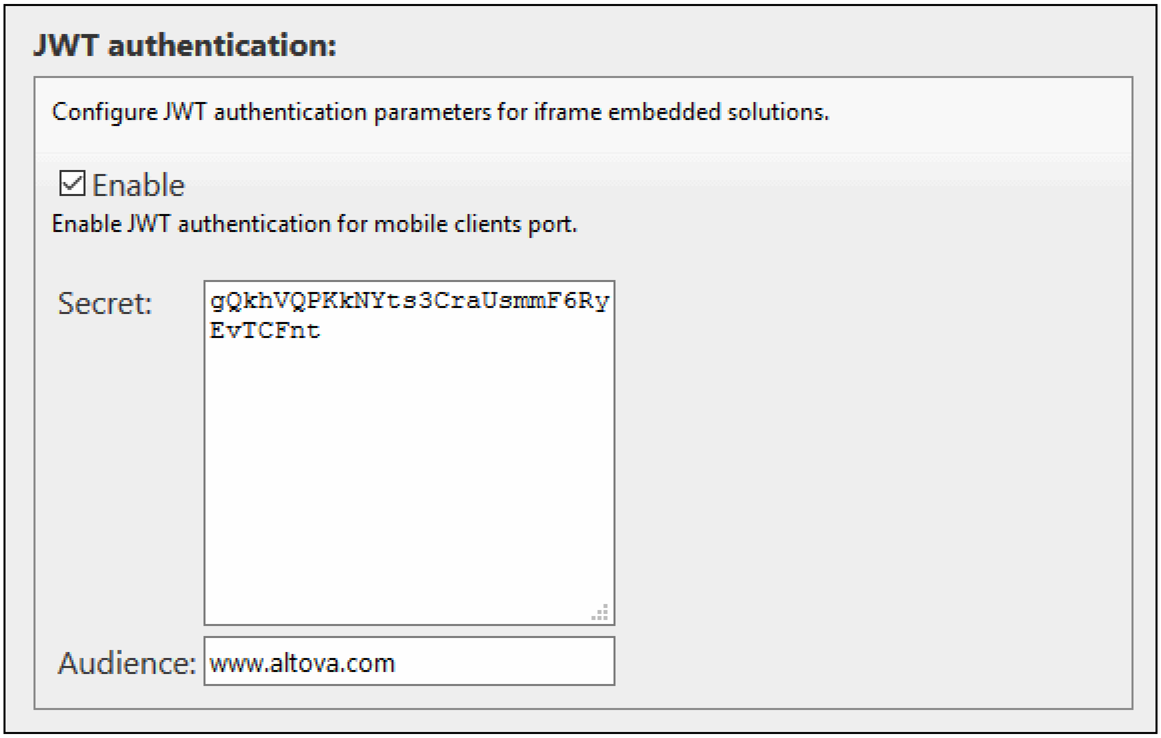
The files used in this example are `JWT.html` and `jsonBooks.mtd`. Both are located in your (72 My) Documents 72 MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions`**. To try out the files, deploy the MTD file to your server, and enable JWT authentication in the server settings (*see next section below*). If **`newuser`** is not registered on your server, it will automatically be imported as a user, and the login will be successful. However, you will need to set permissions so that the container of **`jsonBooks.mtd`** can be accessed. If needed, modify the HTML code so that the correct workflow is targeted.

The JWT in this example file was created with the *Audience* claim set to `www.altova.com` and the *Subject* claim (which specifies the user name) set to `newuser`. The secret used to generate this JWT is `gQkhVQPKkNYts3CraUsmmF6RyEvTCFnt`.

## Server settings

In order for the server to decrypt and verify the JWT sent by the webpage, JWT authentication must be enabled in the server settings (*screenshot below*) with the following two settings:

- The secret used to generate the JWT: `gQkhVQPKkNYts3CraUsmmF6RyEvTCFnt`
- The value of the *Audience* claim that was used to generate the JWT: `www.altova.com`



**JWT authentication:**

Configure JWT authentication parameters for iframe embedded solutions.

☑ Enable

Enable JWT authentication for mobile clients port.

Secret: gQkhVQPKkNYts3CraUsmmF6RyEvTCFnt

Audience: www.altova.com

**Note:** Additionally, remember to set permissions so that the container of **`jsonBooks.mtd`** can be accessed by **`newuser`**.

## HTML code listing

The HTML code listing of the file `JWT.html`. The JWT is highlighted in blue. Please note that some JavaScript functionality used in this example might not be available in all browsers. In this case, please modify the JavaScript to suit your browser.

*Webpage code listing*

```
<!DOCTYPE html>
<html>
   <head>
      <style>
         * {
            font-family: Segoe UI, Tahoma, Arial, Helvetica, sans-serif;
         }
         iframe {
            width: 100%;
            height: 400px;
            border: 2px solid blue;
            border-radius: 5px;
            margin: 10px 0px;
         }
         code {
            font-size: small;
         }
      </style>
      <script src="http://localhost:8083/js/WebAppIFrame.js"></script>
      <script>
         // The initial book list stored in a variable in JSON format
            var books = {
               "books": [
                  {
                     "author": "Mary Shelley",
                     "title": "Frankenstein; or, The Modern Prometheus",
                     "year": 1818,
                     "pages": 280
                  },
                  {
                     "author": "Bram Stoker",
                     "title": "Dracula",
                     "year": 1897,
                     "pages": 302
                  }
               ]
            };

           // Contents of variable 'books' converted to string and displayed inside HTML
element CODE
            function showbooks() {
               document.querySelector('code').innerText = JSON.stringify(books, null, '
');
            }

            // m = HTML message event; data = container for message from server
            // m.data.json = contents of the 'json' object that was sent from the server
            function receivebooks(m) {
               books = m.data.json;
               showbooks();
            }

            // Handler to show books in webpage on page load
            document.addEventListener('DOMContentLoaded', showbooks);

            // Handler to receive messages from server via solution in IFrame
            window.addEventListener('message', receivebooks);

            // Handler to send data to IFrame on page load
            document.addEventListener('DOMContentLoaded', function() {
               var embedded = new WebAppIFrame(document.querySelector('iframe'));
               embedded.start('http://localhost:8083/run?d=/public/jsonBooks', books,
'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJPbmxpbmUgSldUIEJ1aWxkZXIiLCJpYXQiOjE1MDE
4MzU4MzUsImV4cCI6MTUzMzM3MTgzNSwiYXVkIjoid3d3LmFsdG92YS5jb20iLCJzdWIiOiJuZXd1c2VyIiwiR2l2
ZW5OYW1lIjoiSm9obm55IiwiU3VybmFtZSI6IlJvY2tldCIsIkVtYWlsIjoianJvY2tldEBleGFtcGxlLmNvbSIsI
lJvbGUiOlsiTWFuYWdlciIsIlByb2plY3QgQWRtaW5pc3RyYXRvciJdfQ.M66SBrP_U30iQeheQWpPTdaBzsJZqK2
L7BJQ8gKP-Lo');
            });
      </script>
   </head>


   <body>
      <h4>An editable list of books in JSON format</h4>
      <h5>The book list, stored as a JSON object in the webpage:</h5>
      <pre><code><!-- The SHOWBOOKS function displays the book list here --></code></pre>
      <h5>The book list is displayed in the Iframe as soon as the HTML page is
opened.</h5>
      <iframe frameborder="0"></iframe>
   </body>


</html>
```

# 23 AppStore Apps

You can create MobileTogether apps that can be posted to app stores for end users to download. These customized **AppStore Apps** are created in MobileTogether Designer as follows:

1. Design and test the MobileTogether Designer project from which you wish to generate your app. Create the project in the same way as any other MobileTogether Designer design.
2. Generate the program code[1472] of your appstore app from the design via the **File | Generate Program Code for AppStore Apps**[1580] command. Program code can be generated for Android, iOS, Windows Phone, and Windows.
3. Compile the generated program code[1486] to build your appstore app for the respective devices and operating systems.

**Note:** You can also generate program code for AppStore Apps from MobileTogether packages[295].

The program-code generation[1472] and code-compilation[1486] steps are described in the respective sub-sections of this section.

## Difference between AppStore App and solution on MobileTogether Client

An AppStore App is different than a MobileTogether solution.

- A MobileTogether solution is deployed to a MobileTogether Server, and is accessed via a MobileTogether Client. Multiple client devices can access one or more solutions on one or more MobileTogether Servers.
- An AppStore App on the other hand provides a lightweight alternative that directly accesses only one solution, and does not need to be configured in any way to run.

The differences between the solution and the appstore app are laid out in the table below.

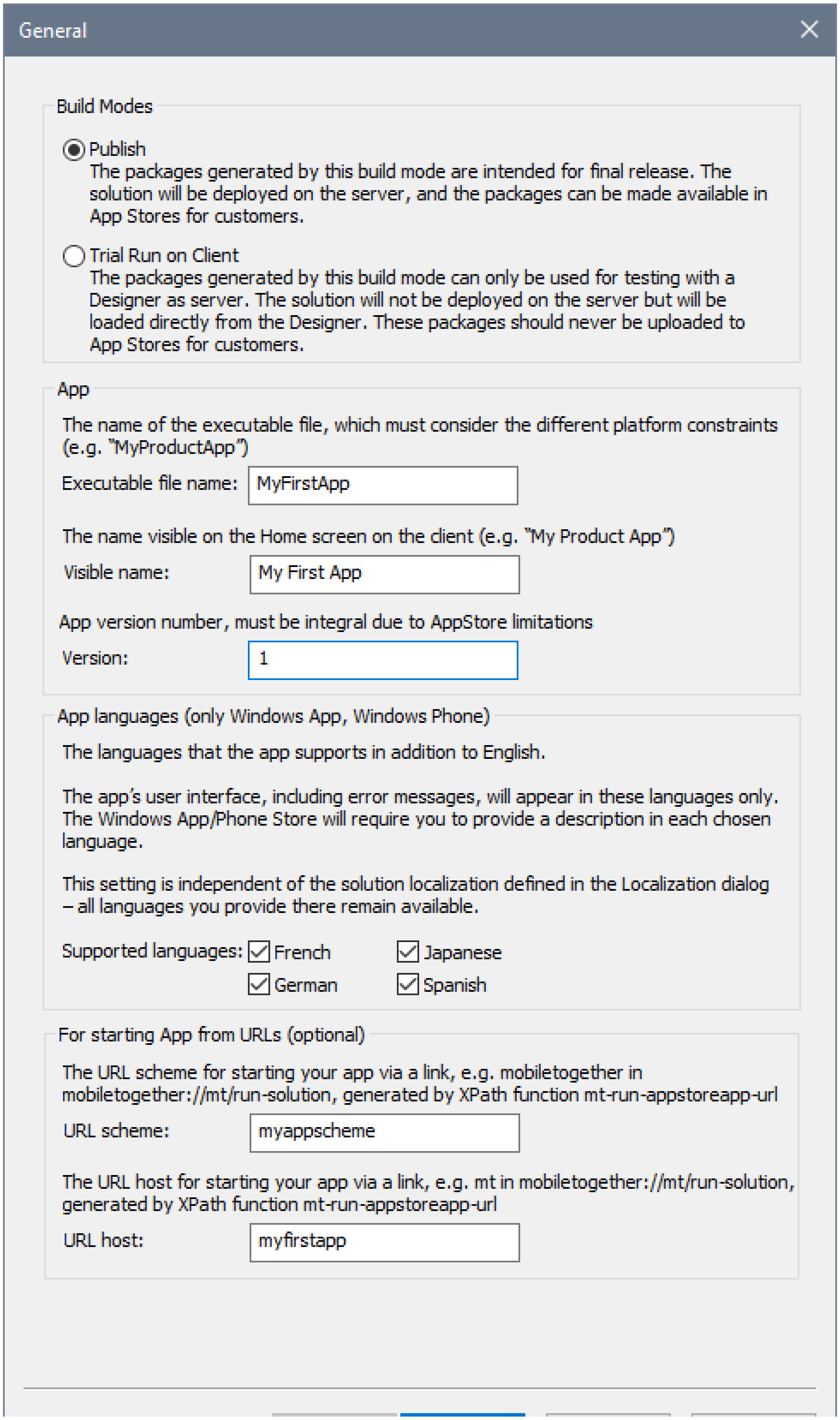
| Solution on MobileTogether Client | AppStore App |
|---|---|
| MobileTogether Designer project is deployed by developer to MobileTogether Server as a solution | MobileTogether Designer project is deployed by developer to MobileTogether Server as an AppStore App solution |
| MobileTogether Client is downloaded by end user from app store | AppStore App is downloaded by end user from app store |
| MobileTogether Client on end-user devices can access solutions on one or more MobileTogether Servers | AppStore App accesses its associated solution. It is "dedicated" to this one solution |
| End user needs to configure and maintain MobileTogether Client. Access to multiple servers and solutions is possible | No MobileTogether Client required. AppStore App provides end user with easy and straightforward access to a single solution |
| From MobileTogether Server, a deployed solution can be opened to run in a web browser. | The deployed AppStore App cannot be opened in a web browser from MobileTogether Server. |

# 23.1 Generate Program Code from Project

To generate program code for appstore apps that run on Android, iOS, Windows Phone, and Windows mobile devices, click the command **File | Generate Program Code for AppStore Apps** 1580. The command opens a wizard that has seven screens if all mobile formats are selected. Each screen contains settings for the generated code.

**Note:** You can also generate program code for AppStore Apps from MobileTogether packages 295.

▼ 1: General: Names, Version, Languages, URL



Build Modes

● Publish
The packages generated by this build mode are intended for final release. The solution will be deployed on the server, and the packages can be made available in App Stores for customers.

○ Trial Run on Client
The packages generated by this build mode can only be used for testing with a Designer as server. The solution will not be deployed on the server but will be loaded directly from the Designer. These packages should never be uploaded to App Stores for customers.

App

The name of the executable file, which must consider the different platform constraints (e.g. "MyProductApp")

Executable file name: MyFirstApp

The name visible on the Home screen on the client (e.g. "My Product App")

Visible name: My First App

App version number, must be integral due to AppStore limitations

Version: 1

App languages (only Windows App, Windows Phone)

The languages that the app supports in addition to English.

The app's user interface, including error messages, will appear in these languages only. The Windows App/Phone Store will require you to provide a description in each chosen language.

This setting is independent of the solution localization defined in the Localization dialog – all languages you provide there remain available.

Supported languages: ☑ French ☑ Japanese ☑ German ☑ Spanish

For starting App from URLs (optional)

The URL scheme for starting your app via a link, e.g. mobiletogether in mobiletogether://mt/run-solution, generated by XPath function mt-run-appstoreapp-url

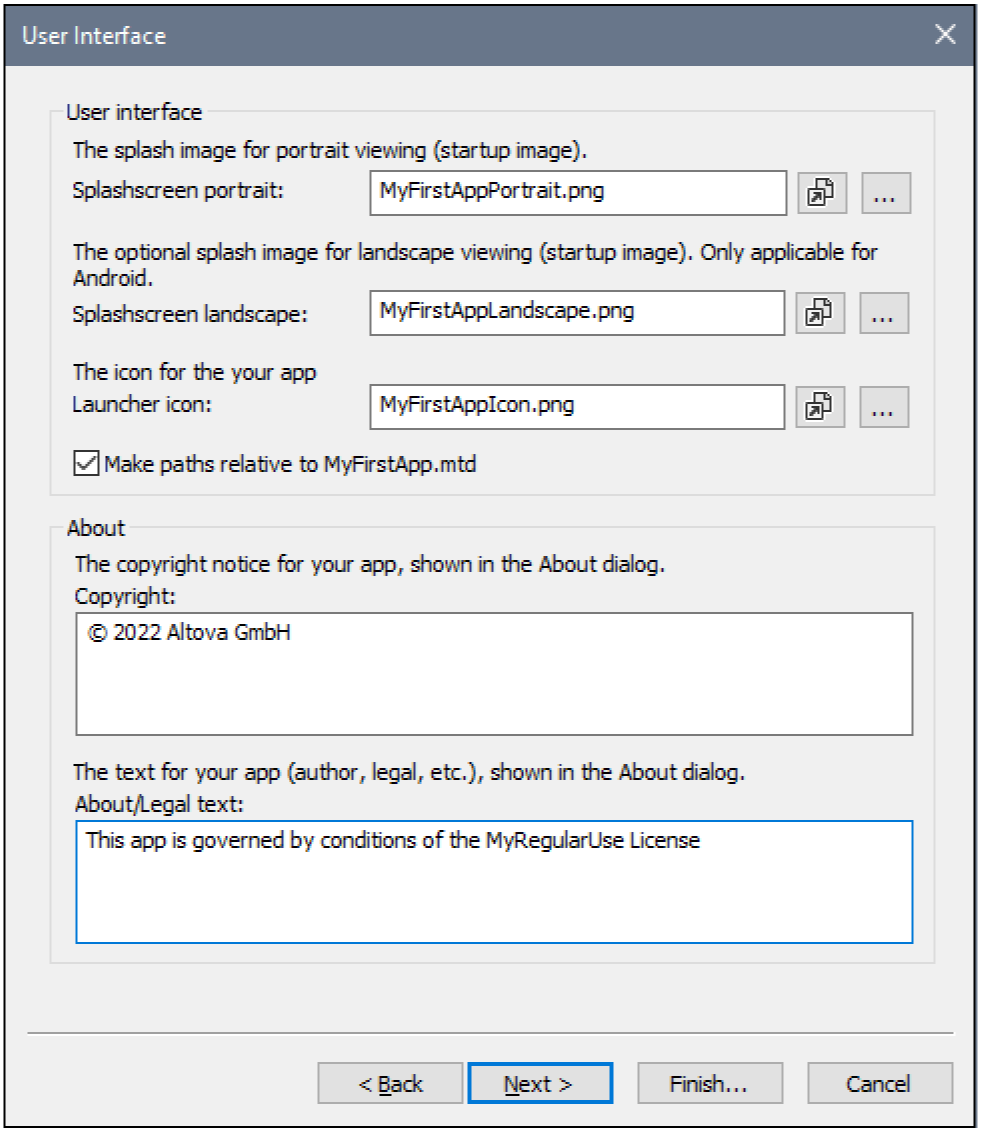
URL scheme: myappscheme

The URL host for starting your app via a link, e.g. mt in mobiletogether://mt/run-solution, generated by XPath function mt-run-appstoreapp-url

URL host: myfirstapp

- *Build modes:* (i) The *Publish* option generates the app for publication to app stores; (ii) The *Trial Run on Client* option generates a package that can be used to test the package by simulating a trial run on the client device 1370. The compiled app will connect to MobileTogether Designer and use the design file in MobileTogether Designer. This means that you would be able to modify the design in (MobileTogether Designer) and directly test the effect of the modifications in a client simulation. Note that if the *Trial Run on Client* option is selected, then *Step 10: Deploy* will be skipped—since the app must not be deployed to the server.
- *Executable file name:* The name that is used internally to reference the code. You should use a name that has no spaces.
- *Visible name:* The name of the app that will be visible to the user.
- *Version:* The version number of the app. Must be an integer or a decimal number. For example: `1` or `1.0` or `1.1` or `1.21`. In case you do not want to deploy for all platforms (for example, because an app that was not approved by one app store has to have its design modified and code regenerated), then the best practice is as follows. Increase the version number by one, and generate program code for the platform/s you want. For example: **`v1.2`** is approved on all platforms but rejected on iOS; **`v1.3`** is created for iOS and approved for iOS (it is not submitted to other stores); **`v1.4`** is created for all platforms and accepted by all stores (so non-iOS jumps from **`v1.2`** to **`v1.4`**)
- *App languages (on Windows App and Windows Phone only):* The app's user interface can be displayed in **`EN`**, **`ES`**, **`FR`**, **`DE`**, **`JA`**. Select the languages you want to include in the app. If the language of the mobile device is one of the selected languages, then the app is displayed in this language. If the language of the mobile device is not among the selected languages, the app defaults to English. Note that the language of the app interface is independent of the localization of solution strings 308.
- *URL scheme and host:* The URL that will start the app from a hyperlink. The hyperlink's target URL will have the format: **`<url-scheme>://<url-host>`**. Enter a unique URL scheme and unique URL host. The scheme information will be stored in the app's manifest file, and indicates to the device that the app can be used to open URLs that start with this scheme. If a link having a URL with this scheme is tapped, then the device will access the resource pointed to by the URL—which is the app.

▼ 2: User Interface: Icons, Copyright, Legal



User Interface

User interface

The splash image for portrait viewing (startup image).

Splashscreen portrait: MyFirstAppPortrait.png

The optional splash image for landscape viewing (startup image). Only applicable for Android.

Splashscreen landscape: MyFirstAppLandscape.png

The icon for the your app

Launcher icon: MyFirstAppIcon.png

☑ Make paths relative to MyFirstApp.mtd

About

The copyright notice for your app, shown in the About dialog.

Copyright:

© 2022 Altova GmbH

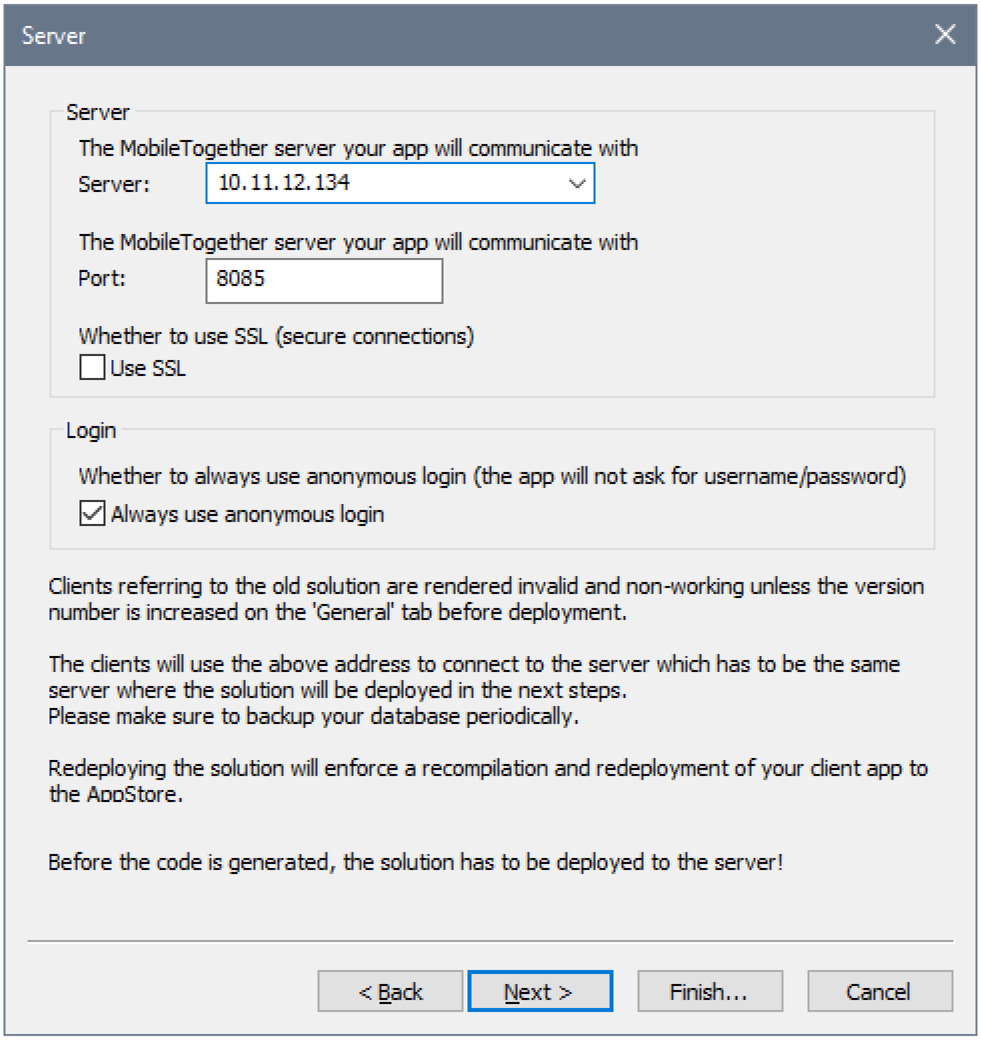
The text for your app (author, legal, etc.), shown in the About dialog.

About/Legal text:

This app is governed by conditions of the MyRegularUse License

- *Splashscreens:* Browse for the splashscreens that should be used in portrait and landscape orientations. The corresponding splashscreen will be used on the mobile device when the device's orientation changes. For iOS apps, the portrait splashscreen is drawn in both orientations to aspect-fill the screen in both orientations. If you want individual splashscreens for orientations/devices, then assign different splashscreens for different dimensions before building the generated program code as described here[1487]. Click the **Open File** icon to view the image file in the system's default image viewer app.
- *Launcher icon:* The icon that is displayed in the *Apps* screen of the mobile device to launch the app. The maximum pixel size of icons is `200x200`. Click the **Open File** icon to view the image file in the system's default image viewer app.
- *Copyright and legal notice:* The text to be displayed in the mobile device.

▼ 3: Server: Server and Login Settings



Server

The MobileTogether server your app will communicate with

Server: 10.11.12.134

The MobileTogether server your app will communicate with

Port: 8085

Whether to use SSL (secure connections)

□ Use SSL

Login

Whether to always use anonymous login (the app will not ask for username/password)

☑ Always use anonymous login

Clients referring to the old solution are rendered invalid and non-working unless the version number is increased on the 'General' tab before deployment.

The clients will use the above address to connect to the server which has to be the same server where the solution will be deployed in the next steps.
Please make sure to backup your database periodically.

Redeploying the solution will enforce a recompilation and redeployment of your client app to the AppStore.

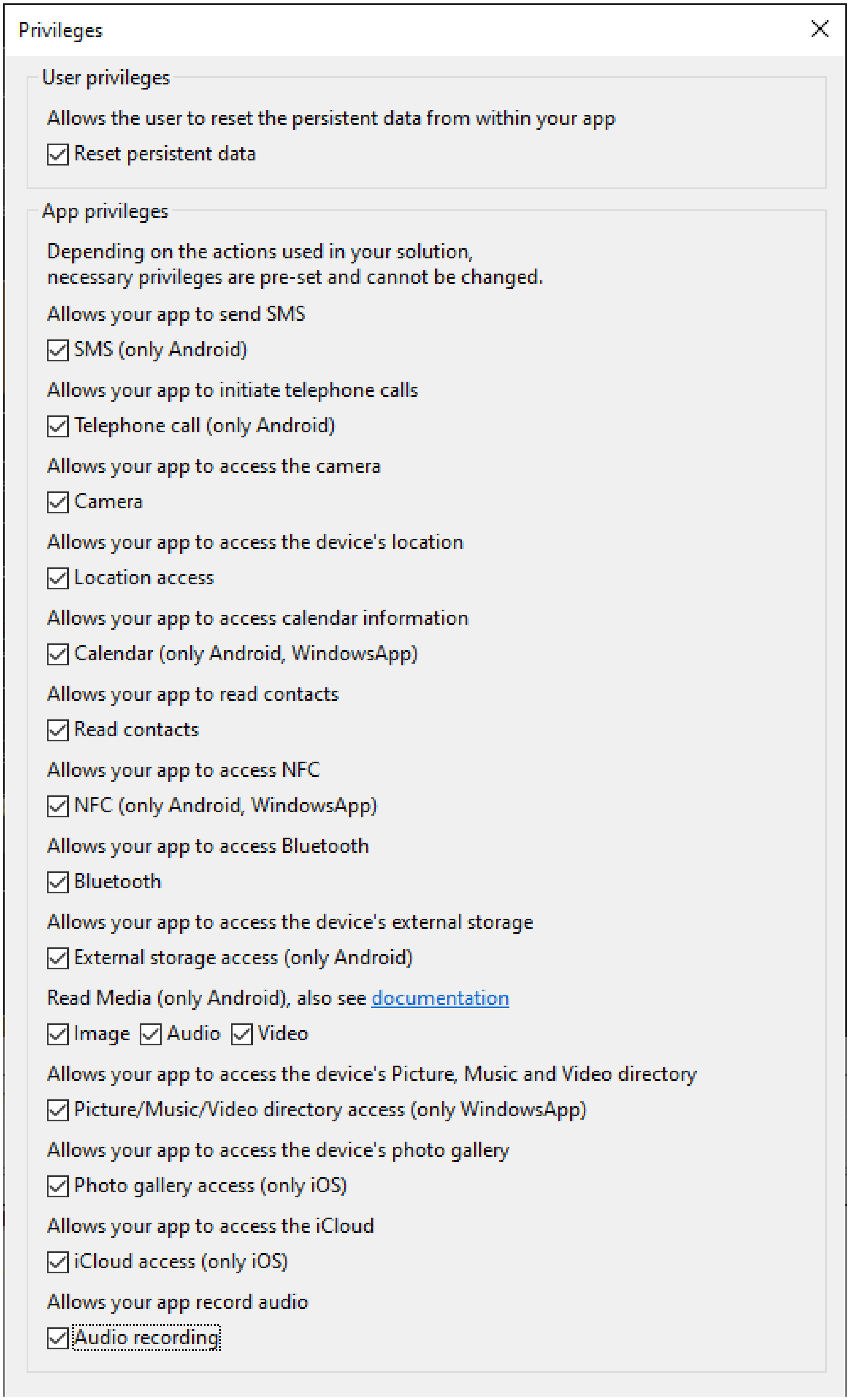
Before the code is generated, the solution has to be deployed to the server!

In Screen 1 you can select (in the *Build Modes* option) whether the appstore app you are currently building is for publication or for simulations on a client device. If you are generating for publication, then specify the connection details of the MobileTogether Server to which the app will connect. If you are generating for simulation on a client, then specify the connection details of the MobileTogether Designer machine. The screenshot above shows Screen 3 when the build is for publication; this is why the first pane takes the properties of the Server rather than those of the Designer.

- *Server/Designer:* The IP address of the server on which the workflow will be deployed (publish mode) or of the designer that you want to use for trial runs (client-simulation mode).
- *Port:* The port on the server/designer via which the app can be accessed. The server's client device access port is set in MobileTogether Server. See the MobileTogether Server user manual for information. For trial runs, you can use the designer as the server. The designer's client device access port is set in the Trial Run on Client tab of the Options dialog 1667. For the *Port* setting, enter the Server's or Designer's device access port.
- *SSL:* If you use SSL, this will need to be set up in MobileTogether Server. See the MobileTogether Server user manual for information.

- *Always use anonymous login:* Select this option if you want to allow end users to access the app without providing login details. Otherwise, end users will need a user name and password to log in. See the MobileTogether Server user manual for information about setting up login credentials.

▼ 4: Privileges: User and App privileges



Privileges

User privileges

Allows the user to reset the persistent data from within your app

☑ Reset persistent data

App privileges

Depending on the actions used in your solution,
necessary privileges are pre-set and cannot be changed.

Allows your app to send SMS

☑ SMS (only Android)

Allows your app to initiate telephone calls

☑ Telephone call (only Android)

Allows your app to access the camera

☑ Camera

Allows your app to access the device's location

☑ Location access

Allows your app to access calendar information

☑ Calendar (only Android, WindowsApp)

Allows your app to read contacts

☑ Read contacts

Allows your app to access NFC

☑ NFC (only Android, WindowsApp)

Allows your app to access Bluetooth

☑ Bluetooth

Allows your app to access the device's external storage

☑ External storage access (only Android)

Read Media (only Android), also see documentation

☑ Image ☑ Audio ☑ Video

Allows your app to access the device's Picture, Music and Video directory

☑ Picture/Music/Video directory access (only WindowsApp)

Allows your app to access the device's photo gallery

☑ Photo gallery access (only iOS)

Allows your app to access the iCloud

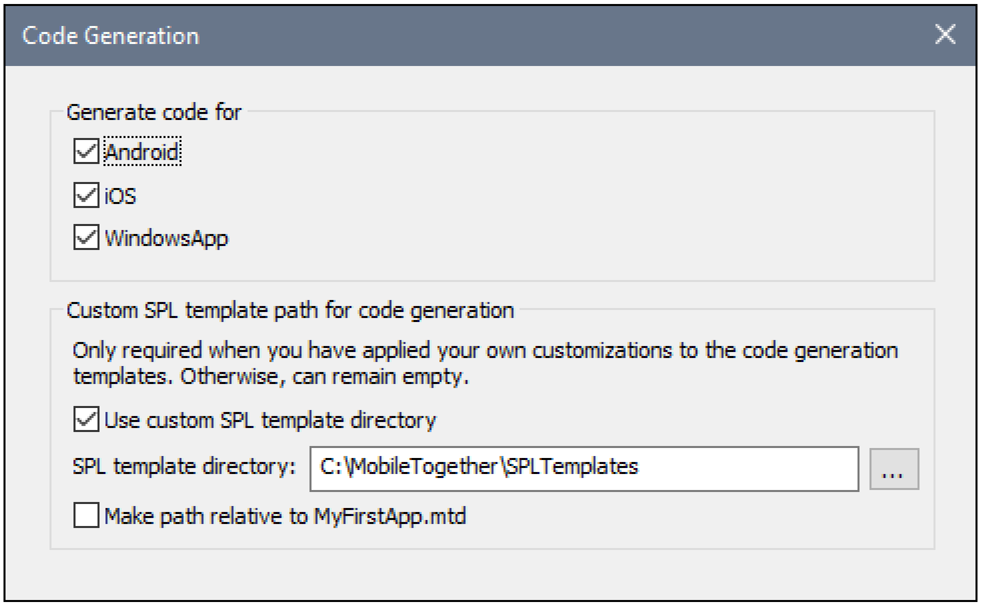
☑ iCloud access (only iOS)

Allows your app record audio

☑ Audio recording

You can set whether the end user is allowed to reset persistent data or not. App privileges are privileges that the mobile device OS grants the app. The privileges you select here are stored in the app's manifest files. When the app is installed, the device checks the app's manifest and informs the end user about the privileges that the app is requesting. If the end user allows these privileges, the app will be installed and the requested privileges are granted to the app. For example, if the design contains a *Send SMS* 698 action, then this privilege will be preset by default and cannot be changed. Location access refers to the GPS location information of the mobile device. In the screenshot above, the design uses geolocation features, and so requires access to the geolocation information of the device. Because of this the *Location access* privilege is automatically selected, and you cannot change it.

## 5: Code Generation: App Formats and SPL Template Location



Code Generation

Generate code for

☑ Android

☑ iOS

☑ WindowsApp

Custom SPL template path for code generation

Only required when you have applied your own customizations to the code generation templates. Otherwise, can remain empty.

☑ Use custom SPL template directory

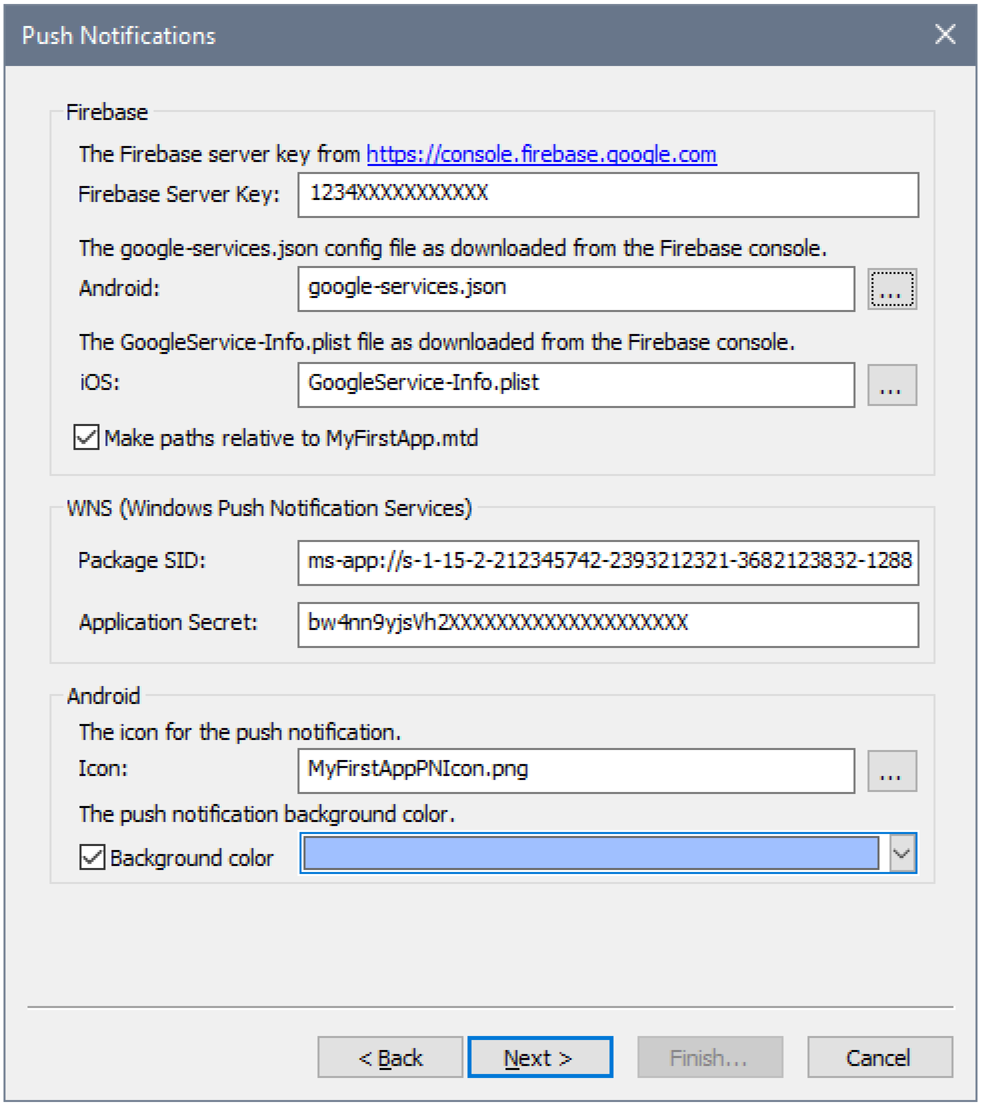
SPL template directory: C:\MobileTogether\SPLTemplates

□ Make path relative to MyFirstApp.mtd

Select the app formats for which you want to generate program code. SPL templates for the different app formats are required for generation of the program code. These are provided with your MobileTogether Designer installation and are located here: `C:\Program Files (x86)\Altova\MobileTogetherDesigner10\MobileTogetherSPL`. This directory will be the default SPL template directory.

If you customize one or more SPL templates, create a copy of the SPL template directory (in which the customized files will be saved). Specify its location of this directory in the *SPL template directory* option of this screen (*see screenshot above*). The new SPL template directory must reproduce the directory structure of the original. The option *Use custom SPL template directory* enables you to switch quickly between the original SPL template directory and your custom SPL template directory. Select or deselect the option according to which templates you want to use (custom or original).

## 6: Push Notifications



Push Notifications

Firebase

The Firebase server key from https://console.firebase.google.com

Firebase Server Key: 1234XXXXXXXXXXX

The google-services.json config file as downloaded from the Firebase console.

Android: google-services.json

The GoogleService-Info.plist file as downloaded from the Firebase console.

iOS: GoogleService-Info.plist

☑ Make paths relative to MyFirstApp.mtd

WNS (Windows Push Notification Services)

Package SID: ms-app://s-1-15-2-212345742-2393212321-3682123832-1288

Application Secret: bw4nn9yjsVh2XXXXXXXXXXXXXXXXXXXX

Android

The icon for the push notification.

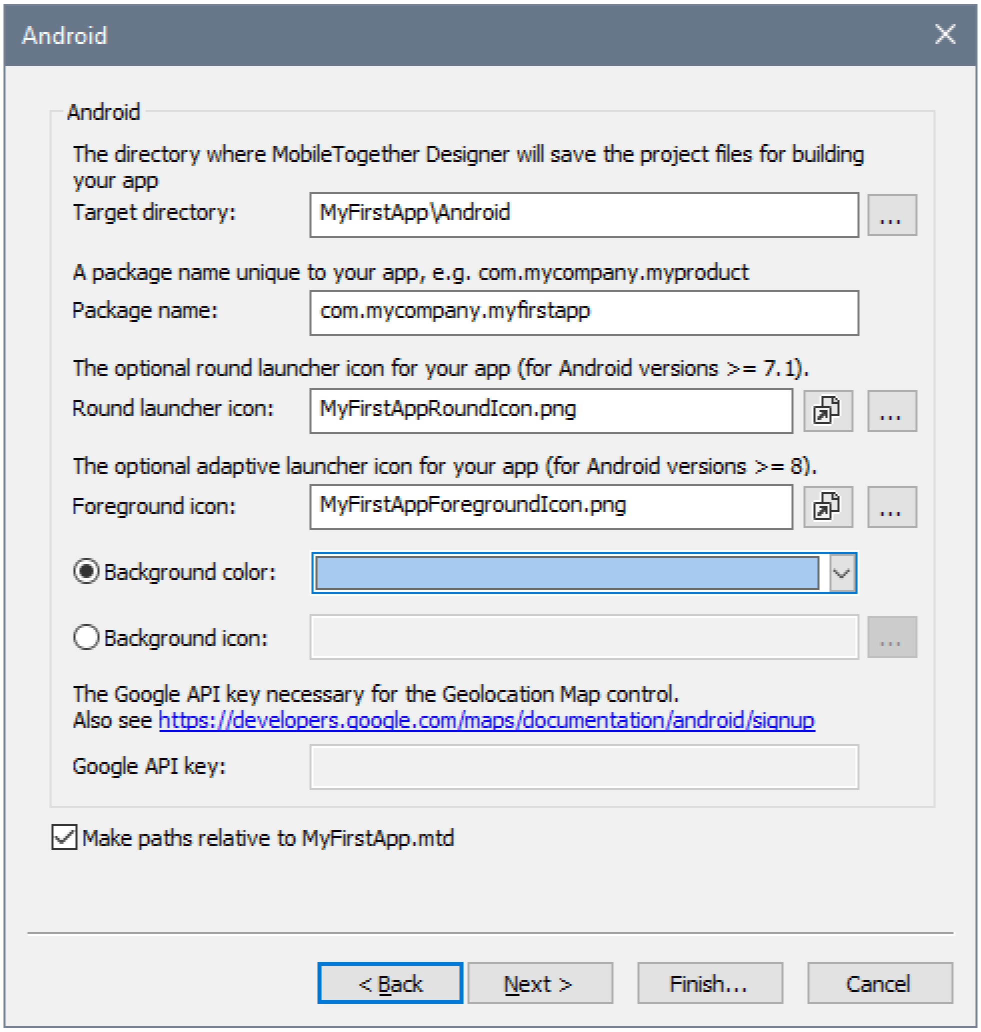
Icon: MyFirstAppPNIcon.png

The push notification background color.

☑ Background color

This screen appears only if the **OnPushNotificationReceived**[296] event has an action defined for it. In this screen, enter the details you obtained when making the push notification (PN) registrations for the different operating systems (*see Push Notifications in AppStore Apps*[1130])[1130]. For PNs that are sent to Android devices, you can also select a background color for the PN.

▼ 7: Android



The directory where MobileTogether Designer will save the project files for building your app

Target directory: MyFirstApp\Android

A package name unique to your app, e.g. com.mycompany.myproduct

Package name: com.mycompany.myfirstapp

The optional round launcher icon for your app (for Android versions >= 7.1).

Round launcher icon: MyFirstAppRoundIcon.png

The optional adaptive launcher icon for your app (for Android versions >= 8).

Foreground icon: MyFirstAppForegroundIcon.png

● Background color:

○ Background icon:

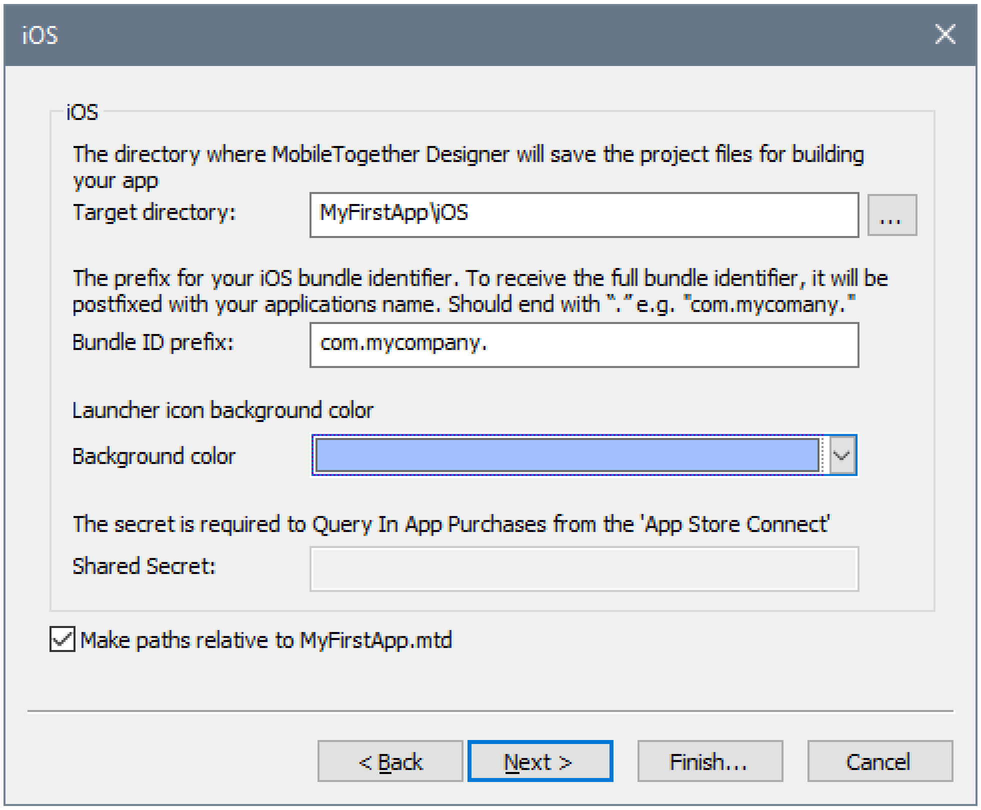
The Google API key necessary for the Geolocation Map control.
Also see https://developers.google.com/maps/documentation/android/signup

Google API key:

☑ Make paths relative to MyFirstApp.mtd

Sets the target directory where the program code will be generated for the Android format. You must specify the package name for the Android package. You can also select icons for Android's round launcher and adaptive launcher, as well as the background color of the adaptive launcher. To preview the image files, click the respective image's **Open File** icon. The *Google API Key* field is enabled if the design contains a Geolocation Map[508] control. A Google API Key is needed for enabling the features of the control. You can get an API key via your developer account on the Google Cloud Platform.

8: iOS



iOS

iOS

The directory where MobileTogether Designer will save the project files for building your app

Target directory: MyFirstApp\iOS

The prefix for your iOS bundle identifier. To receive the full bundle identifier, it will be postfixed with your applications name. Should end with "." e.g. "com.mycomany."

Bundle ID prefix: com.mycompany.

Launcher icon background color

Background color

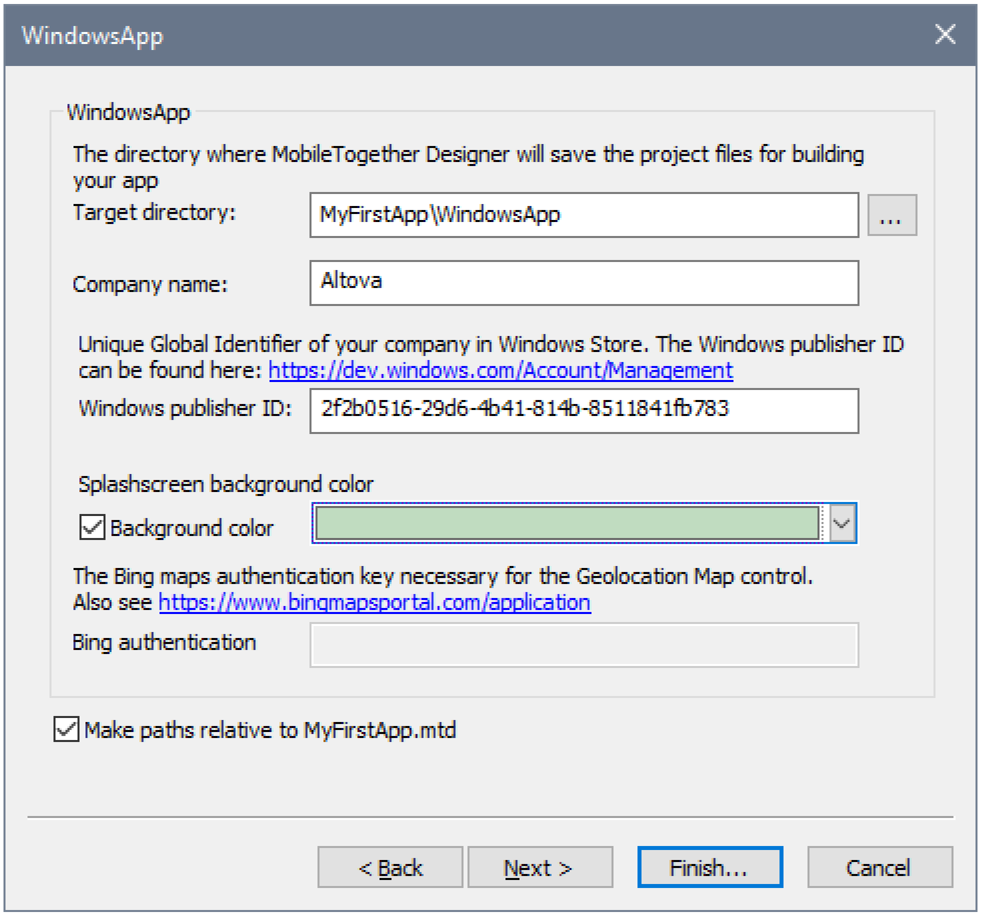
The secret is required to Query In App Purchases from the 'App Store Connect'

Shared Secret:

☑ Make paths relative to MyFirstApp.mtd

Sets the target directory where the program code will be generated for the iOS format. You must specify the Bundle ID prefix for iOS. Make sure that the Bundle ID prefix ends with the dot character `.`. For example, `com.altova.` has the required format. The Bundle ID is constructed by appending the app name you provided in Screen 1 to the Bundle ID prefix. If you have specified that the app should have iCloud access, then an **iCloud Container ID** is automatically generated and stored in a file called **`<appname>.entitlements`**. This file will be created automatically in the target directory of the program code. (*See Compile Program Code: iOS[1487] for more information.)* You can also set the background color of the iOS launcher icon. For In-App purchases, you can set the shared secret of the app at the App Store.

▼ 9: WindowsApp



WindowsApp

WindowsApp

The directory where MobileTogether Designer will save the project files for building your app

Target directory: MyFirstApp\WindowsApp

Company name: Altova

Unique Global Identifier of your company in Windows Store. The Windows publisher ID can be found here: https://dev.windows.com/Account/Management

Windows publisher ID: 2f2b0516-29d6-4b41-814b-8511841fb783

Splashscreen background color

☑ Background color 🟩

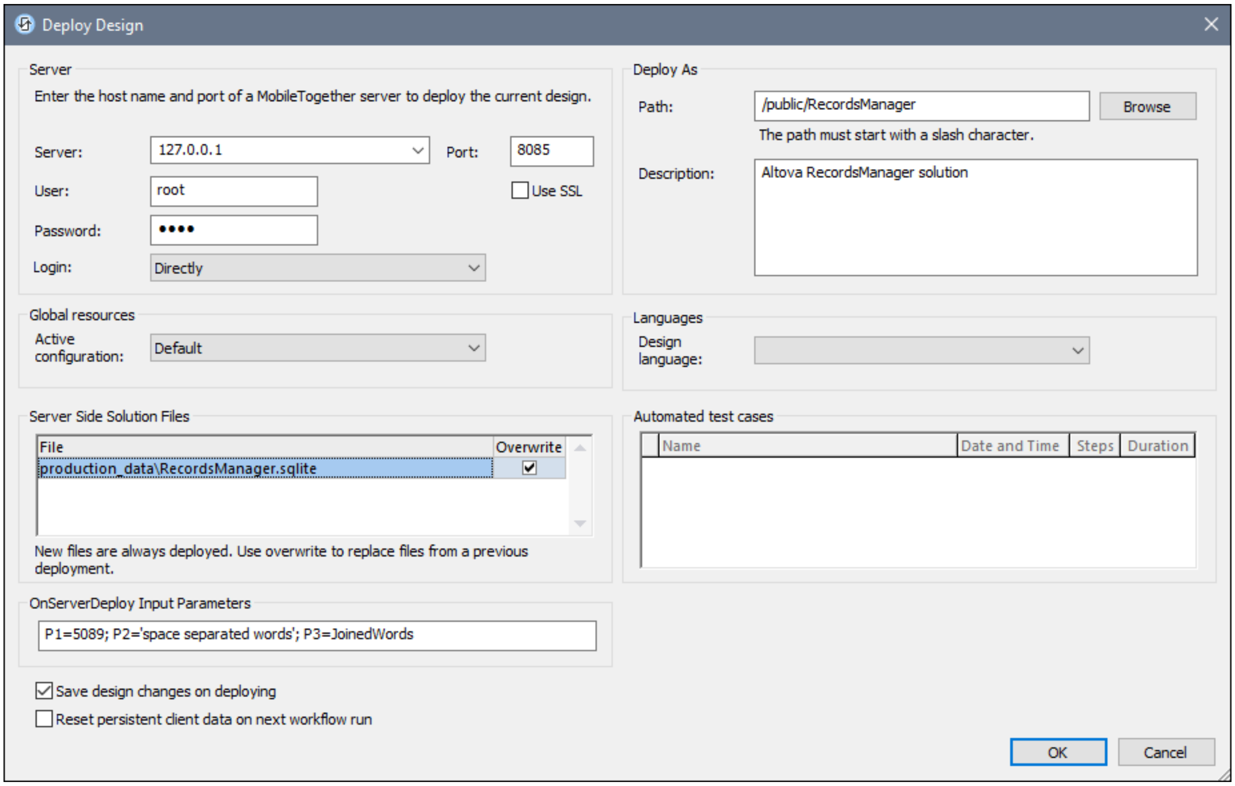
The Bing maps authentication key necessary for the Geolocation Map control. Also see https://www.bingmapsportal.com/application

Bing authentication

☑ Make paths relative to MyFirstApp.mtd

Sets the target directory where the program code will be generated for the Windows App format. You must specify your company's Windows Publisher ID. The Publisher ID is the GUID that’s assigned to your developer account (*see Windows App | Requirements* 1489; your Windows Publisher ID can be found on your Account Summary page in the Dev Center; the link in the dialog takes you there). You can also select a background color for the splashscreen of the app. The *Bing Authentication* field is enabled if the design contains a Geolocation Map 508 control. A Bing Authentication Key is needed for enabling the features of the control. You can get an authentication key after registering as a developer at the Bing Maps Dev Center.

▼ 10: Deploy



Deploy Design

Server

Enter the host name and port of a MobileTogether server to deploy the current design.

Server: 127.0.0.1 Port: 8085

User: root □ Use SSL

Password: ••••

Login: Directly

Global resources

Active configuration: Default

Server Side Solution Files

| File | Overwrite |
|---|---|
| production_data\RecordsManager.sqlite | ☑ |

New files are always deployed. Use overwrite to replace files from a previous deployment.

OnServerDeploy Input Parameters

P1=5089; P2='space separated words'; P3=JoinedWords

☑ Save design changes on deploying

□ Reset persistent client data on next workflow run

Deploy As

Path: /public/RecordsManager

The path must start with a slash character.

Description: Altova RecordsManager solution

Languages

Design language:

Automated test cases

| Name | Date and Time | Steps | Duration |
|---|---|---|---|

Fill in the data fields of the dialog as described in the description of the **Deploy to MobileTogether Server**[1574] command.

On clicking **OK**, the following happens:

1. The workflow is deployed to MobileTogether Server.
2. Program code is generated for all the selected formats in the respective target directories that you specified.

You can now compile the generated program code[1486] for the respective formats (Android, iOS, Windows App and/or Windows Phone) into the respective AppStore Apps.

## The workflow key

Each time program code is generated and the workflow is deployed to the server, a unique **workflow key** is assigned to the program code and to the workflow on the server. When the program code is compiled, the workflow key is stored in each of the compiled formats. This workflow key serves as the "handshake" that associates a compiled app with a particular workflow and allows the app to access the workflow.

If you run the Generate Program Code Wizard[1472] again, the program code will be re-generated and the workflow will be re-deployed to the server. The app/s and the workflow will have a new workflow key. If the newly deployed workflow overwrites the previous workflow, then previous app versions will no longer be able to access the new workflow on the server. This is because the workflow key of the previous app/s will not match the workflow key of the newly deployed workflow. Only apps generated from the new program code will be able to access the newly deployed workflow (since both have the same workflow key).

Therefore, every time you modify the solution, assign **a new version number** (in Screen 1 of the Generate Program Code Wizard 1472) before generating the code and deploying the workflow to the server. This way, the previous workflow on the server is not replaced by the new workflow. Apps of the previous version can continue to use the previous workflow, while apps of the new version can use the new workflow.

**Note:** Even if you have not changed your design, a new unique workflow key is generated every time you run the Generate Program Code Wizard 1472.

**Note:** To ensure that you do not lose compatibility between apps and workflows, we recommend that you back up your MobileTogether Server periodically. This is so that you do not lose previously deployed workflows. See the MobileTogether Server manual for information about backing up MobileTogether Server.

# 23.2 Compile Program Code

After you complete the Generate Program Code Wizard 1472 and click **OK** in the Deploy Design 1472 screen, the app's workflow will be deployed to the server and program code for the selected app formats will be generated. Program code is generated for each app format separately, and is generated in the respective target directories that you specified in the wizard 1472. This section describes how to compile the respective program codes into the AppStore Apps that you can upload to the respective app stores.

The program code that is generated by MobileTogether Designer does not usually need modifying before it is built. However, if you need to modify the project, it is important that you modify the SPL template files instead of the generated project files. This is so that your changes are not lost when you subsequently generate the project via the Generate Program Code Wizard 1472. SPL and SPL templates are described in the section SPL Templates 1492.

Program code is generated for the app formats listed below. The links below take you to the respective sub-sections. Each sub-section describes how to compile the program code for that app format.

- Android 1486
- iOS 1487
- Windows App 1489

## 23.2.1 Android

When you complete the wizard, MobileTogether Designer creates an Android Studio project. You can use Android Studio to build this project into an Android app (which is a **`.apk`** file).

### The Android Studio project

The Android Studio project is created in the directory you selected as the program code's destination folder (in Screen 6 of the Generate Program Code Wizard 1472).

Given below are the locations of key project files. These files do not need to be changed before they are compiled in Android Studio. But you can customize their content 1492 should the need arise.

**`app\src\main\AndroidManifest.xml`**

This is the Android manifest. It contains app-related information—such as package name and app version—that you entered in the Generate Program Code Wizard 1472.

**`app\src\main\java\<package-name>\MainActivity.java`**
**`app\src\main\java\<binary-name>.java`**

The values of <*`package-name`*> and <*`binary-name`*> are taken from the values you entered in Screen 6 and Screen 1, respectively, of the Generate Program Code Wizard 1472.

The directory **`app\src\main\res\`** contains resource files for the app icon, splashscreens, and miscellaneous resource strings. The app icon is available in various sizes in the appropriate subdirectories. The various sizes are created automatically by MobileTogether Designer from the icon file you specify in Screen 2 of the Generate Program Code Wizard 1472.

### Building the Android app

The app will be built with Android Studio. The steps described here apply to Android Studio 3.3, which is the version that is current at the time of writing (2019).

Open the project (the target directory) in Android Studio. The build will start automatically, and the result is shown in the Gradle Console. During the build process, you might receive the message: `AAPT err […] libpng warning: iCCP: Not recognizing known sRGB profile that has been edited`. This can be safely ignored. The APK file is not created immediately. You must run the app in order to create the APK file.

### Running and debugging the app

When the build is complete, the app can be run by issuing one of the following Android Studio commands: **Run <app>** or **Debug <app>**. The APK file is created in: **`app\build\outputs\apk\debug\<app>-debug.apk`**.

### Releasing the Android app

In order to publish the app, you will need to create a signed APK. Do this as follows:

1. In Android Studio, issue the command **Build | Generate Signed APK**. This starts the Generate Signed APK Wizard.
2. Type in the keystore path. You might need to create a new keystore first.
3. Go through the wizard's screens and fill in the information that is required.
4. Click **Finish** to start generating the signed APK.

The signed APK file will be placed in: **`app\release\<app>-release.apk`**. (If Debug was selected, the app will be placed in **`app\debug\<app>-debug.apk`**.)

**Note:** The **Generate Signed APK** command can cause Android Studio to erroneously report `lint` errors in open code files. If this occurs, issuing the command **File | Invalidate Caches / Restart** might remove the errors. If you receive the message: `AAPT err […] libpng warning: iCCP: Not recognizing known sRGB profile that has been edited`, you can safely ignore it.

## 23.2.2 iOS

The generated program code for iOS apps is an XCode project. It is saved in the directory you specified in Screen 6 of the Generate Program Code Wizard[1472]. You must now use XCode to open the **`.xcodeproj`** file and build the iOS app.

### Requirements

In order to build, test and publish the iOS App from the generated program code, you will need the following:

- Latest version of XCode, available from Apple's Developer Platform.
- iOS 13.0 or higher for testing the app.
- Membership in the Apple Developer Program. You will need this to test and publish your app to the App Store.

## Building and distributing the app

The broad outline for building the app is given below. For a complete description, see Apple's App Distribution Guidelines

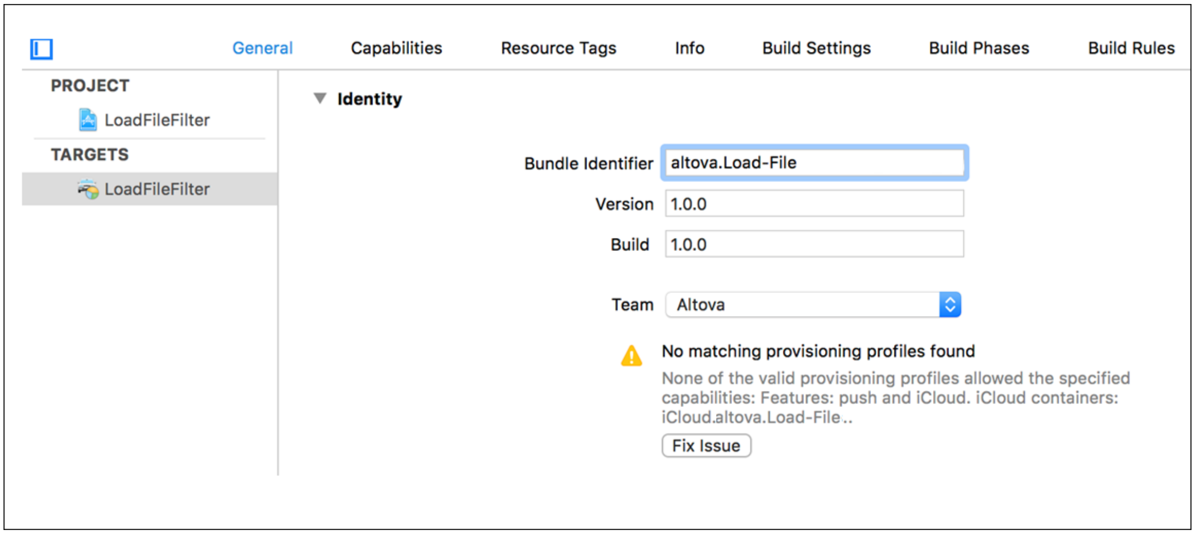
1. Open the `.xcodeproj` file in XCode.
2. Ensure that team, app id, and provisioning profiles are set up. See Configuring Your XCode Project for Distribution for information about how to do this. Note that test devices for ad hoc distribution are only active after re-generating the corresponding provisioning files.
3. Finalize what splashscreen/s to use. By default, the portrait splashscreen (specified in Screen 2 of the Generate Program Code Wizard 1472) will be drawn in both orientations to fill the screen with the same aspect ratio as the original. If, however, you wish to set up individual splashscreens specific to different devices and orientations, do the following: (i) Go to **Supporting Files | Info.plist** and delete *Launch screen interface file base name*; (ii) Click `images.xcassets`, then right-click in the view and choose *New Launch Image*; (iii) Specify a launch image for each resolution and orientation.
4. If you have hooked up an iOS device, go to **Product | Destination** and check your device. Otherwise, go to **Product | Destination** and select *iOS Device*.
5. To test the compiled app on your iOS device, click **Product | Run** (after having selected your iOS device under **Product | Destination**).
6. For ad hoc and App Store submissions, click **Product | Archive** and export your app. Please follow Apple's App Distribution Guidelines.

**Note:** In iOS apps that have multiple pages, when the **Submit** button on the app's last page is tapped, the app will be re-started. This is because an iOS app cannot shut itself down.

## Additional steps for using iCloud

If your app uses iCloud—typically, this happens if your app defines that files are saved to the client—then a few additional steps are required:

1. In XCode, click the *General* tab (*see screenshot below*). You will see a message saying that a provisioning file is missing.



Bundle Identifier: altova.Load-File
Version: 1.0.0
Build: 1.0.0
Team: Altova

No matching provisioning profiles found
None of the valid provisioning profiles allowed the specified capabilities: Features: push and iCloud. iCloud containers: iCloud.altova.Load-File..

2. If you have sufficient developer-account rights, click **Fix Issue**.

3. Click the *Capabilities* tab (*refer to the screenshot above*).
4. In the *Services* section, select *iCloud Documents*.
5. Modify the default settings if required, then click **Fix Issue**.

The steps above provide a quick way, if you have sufficient developer-account rights, to set up an app id and an iCloud container.

An **iCloud Container ID** is required to enable iCloud access for the app. During the generation of program code, an **iCloud Container ID** is automatically generated from the Bundle ID that you specified in Screen 6 of the Generate Program Code Wizard 1472. The generated ID has the form: **`iCloud.`*`<BundleIDPrefix>`*`.`*`<PackageName>`***. It is stored in a file called ***`<appname>`*`.entitlements`**, which is automatically created in the target directory of the program code. If you wish to change the automatically generated ID, modify its name in ***`<appname>`*`.entitlements`**, and save the file. The ID in the Entitlements file must match the iCloud Container ID in your developer account.

For more detailed information about iCloud use, see the following topics in the iOS Developer documentation: Adding Capabilities and Enabling CloudKit in your app.

## 23.2.3 Windows App

The Windows App format is for Windows touch devices (such as tablets running Windows RT) and PCs running Windows 8.1 or higher. The generated program code for Windows Apps is in C++. It is generated in the directory you specified in Screen 7 of the Generate Program Code Wizard 1472. You can open the generated C++ project (`some-app-name.vcxproj`) in Visual Studio and build the project into a Windows App. The output of the build process includes a file with the **`.XAP`** extension. This is a zip file that contains all the files required by your app, and it is the file that you publish to the Windows App Store.

For additional information, see the **`Readme.docx`** file in the MobileTogether Designer program files folder: **`C:\Program Files (x86)\Altova\MobileTogetherDesigner10\MobileTogetherSPL\WindowsApp`**.

### Requirements

In order to build, test and publish the Windows App from the generated program code, you will need the following:

- A Windows app developer account. You will need this to publish your app to the Windows Store. For details, see the Microsoft information about opening an account.
- Visual Studio 2015 Update 3, Visual Studio 2017, and Visual Studio 2019. For each of these versions, the respective additional components listed below are also required.
- Windows 8.1 or higher for testing the app.

Additional components for different Visual Studio versions:

*Visual Studio 2015 Update 3*

- Universal Windows App Development Tools (1.4.1) and Windows 10 SDK (10.0.14393)
- Windows 8.1 and Windows Phone 8.0/8.1 Tools: Tools and Windows SDKs

*Visual Studio 2017*

- C++ Universal Windows Platform Tools

- Windows 10 SDK (10.0.14393.0)
- Windows 8.1 SDK (*can be found under "SDKs, libraries, and frameworks" in "Individual components"*)

*Visual Studio 2019*

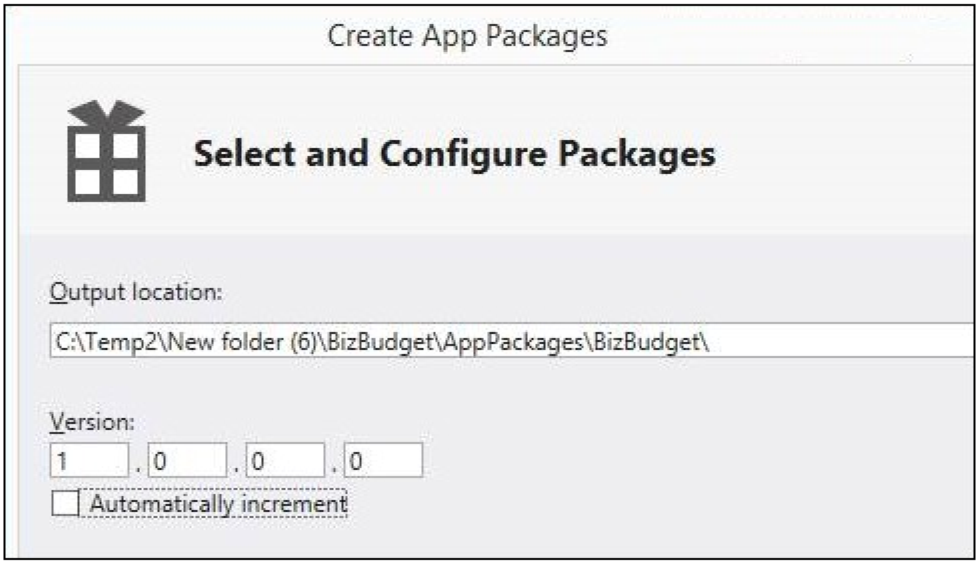
Either one of the following:

- Visual Studio has the following components: (i) Windows 10 SDK (10.0.17134.0); (ii) MSVC v141 – VS 2017 C++ x64/x86 build tools; (iii) MSVC v141 – VS 2017 C++ ARM build tools;
- The generated project is re-targeted to use Toolset v142. Do this as follows: In Visual Studio's Solution Explorer: (i) select the project; (ii) run the **Retarget Projects** command in the context menu. If you expect your app to be used on Windows Phone, then select Windows SDK 10.0.17134.0 *or older*. Note that version 10.0.17763.0 or newer will generate a **.msix** installer package, which cannot be used on Windows Phone. Note also that the minimum version of the Windows SDK that the MobileTogether libraries support is 10.0.10586.0.

## Building the app

The broad outline for building the app is as follows:

1. Open the generated program code in Visual Studio. The file to open will be a **.vcxproj** file in the target directory you specified in Screen 7 of the Generate Program Code Wizard 1472.
2. In Visual Studio, set the build configurations to Windows 32, Windows 64, or ARM. For the *Version* setting (*see screenshot below*), use the same version number as you entered in Screen 1 of the Generate Program Code Wizard 1472. For example: If you entered **10** as the version number in the wizard, then make sure that the version number in Visual Studio is **10.0.0.0**. Also, in Visual Studio, make sure that you uncheck the *Automatically Increment* option of the *Version* setting (*see screenshot below*). If this option is checked, the version number will automatically increment each time you build the app, and will lead to a mismatch with the number entered in the wizard.



Output location:
C:\Temp2\New folder (6)\BizBudget\AppPackages\BizBudget\

Version:
1 . 0 . 0 . 0

☐ Automatically increment

3. In the combo box to the right of the **Run** button (in the toolbar), select **Debug**, and then click **Run**, or press **F5**, or select **Debug | Start Debugging**.
4. Some exceptions will be reported because Visual Studio cannot check the existence of some MobileTogether project or workflow files. Ignore these exceptions.
5. If you need to modify the program code, decide whether the SPL templates that generate the program code need to be edited or whether the modifications are limited to entries in the Generate Program

Code Wizard[1472]. If the former, then edit the SPL template/s. Run the Generate Program Code Wizard[1472] to re-generate the program code, then re-test.

6. To build the release app, go to the combo box next to the **Run** button and select **Release**. Then choose **Project | Store | Create App Packages**. This opens a wizard.
7. In the screens of the wizard, select the options you want and provide the information requested.
8. After completing the wizard, press **Create**. Visual Studio will start building the app.

For each selected platform (Win-32, Win-64, and ARM), an `APPXUPLOAD` file is created. This is the file to upload to the Windows Store. Additionally, for each platform, a folder will be created (with `WindowsStore` in its name), which contains an installer package that enables you to install the compiled app on the respective platform. This enables the app to be tested on multiple machines before it is published to the Windows Store. How to install from this package is described in the next section.

## Installing the app directly

You can install the Windows app directly on PCs or tablets running Windows 8.1 or higher. Installing in this way (as opposed to downloading the app from the Windows Store) is convenient, for example, if you wish to test the app on multiple workstations or distribute it directly.

1. On the PC or tablet, go to the Start page, and search for Windows PowerShell. Right-click it and choose **Run as administrator**.
2. In PowerShell, type: **`set-executionpolicy unrestricted`**
3. Press **Enter**, and confirm with **`Y`** and **Enter**. (You only have to do this once.)
4. Type: **`Show-WindowsDeveloperLicenseRegistration`** , and press **Enter**.
5. Complete the dialog in order to get a Windows Developer License. (You only have to do this once.)
6. Copy the files of the compiled App package to your PC or tablet.
7. In PowerShell, navigate (using the **`cd`** command) to where you copied the files.
8. Type: **`Add-AppDevPackage.ps1`** to start the **`.ps1`** script in that folder, and then press **Enter**. (Another way to start the **`.ps1`** script is to use the context menu command **Run in PowerShell** (or something similar).)
9. Your app should then be installed on the Start page and be ready for testing.

# 23.3 SPL Templates

MobileTogether Designer uses Spy Programming Language (SPL) templates to generate the various files of the program code that is used to build AppStore Apps for Android, iOS, and Windows.

- The SPL templates are delivered with your installation of MobileTogether Designer and are located in the folder **`C:\Program Files (x86)\Altova\MobileTogetherDesigner10\MobileTogetherSPL`**.
- The SPL templates use variables that are tied to information that you enter in the Generate Program Code Wizard 1472.

So, if you need to modify the program code in any way, you should **not** modify the generated code directly. Instead, you should modify the SPL templates that generate the code. This way your modifications will not be lost if you were to re-generate program code from the templates. This section explains how the SPL templates work and what you can edit in them.

## Location of the SPL templates

The SPL templates are located in the MobileTogether Designer application folder:

**`C:\Program Files (x86)\Altova\MobileTogetherDesigner10\MobileTogetherSPL`**

This folder contains entry-point SPL templates for each app format (`Android.spl`, `iOS.spl`, `WindowsApp.spl`, and `WindowsPhone.spl`), a common library SPL template (`CommonLibrary.spl`, which must not be modified), and a folder for each app format that contains additional SPL templates for that format. Inside the folder for each app format is a ZIP file and the SPL template files for that app format.

Each SPL file is a template that can generate a number of project files and call other SPL files. When creating an Android Studio project, for example, MobileTogether Designer begins by processing `Android.spl` (in the SPL template directory). This SPL file in turn calls other SPL files, which then generate other project files and call other SPL files. These SPL template files are the files that you can modify to alter the generated program code. They are written using SPL syntax and SPL instructions, which are described in the sub-sections of this section.

## Using the SPL templates to modify program code

The steps to customize generated program code are as follows:

1. Copy either the entire SPL template directory (or only the templates you wish to modify) to a directory of your choice. In this way a re-installation of MobileTogether Designer will not overwrite your customized SPL templates. Note that the copy of the directory must have the same structure as the original directory.
2. Modify the SPL files as required.
3. Run the Generate Program Code Wizard 1472 with the modified SPL templates. In Screen 5 of the Generate Program Code Wizard 1472, specify the location of the directory containing your customized SPL templates.

The program code will be generated according to your customized templates.

## In this section

This section is organized as follows:

## 23.3.1 SPL Syntax

An SPL template is constructed in the programming language of the program code you wish to generate. The template contains snippets of SPL instructions to integrate MobileTogether data into the generated program code. SPL instructions are enclosed in square brackets. Here, for example, is a template to generate an XML file (written in XML), with the SPL instructions highlighted in yellow.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="[=$Options.androidPackageName]"
    android:versionCode="[=$Options.appVersion]"
    android:versionName="[=$Options.appVersion]">
    <uses-sdk
        android:minSdkVersion="15"
        android:targetSdkVersion="22"/>
</manifest>
```

### Multiple statements

Multiple statements can be included in a bracket pair. Additional statements have to be separated from the previous statement by a new line or a colon. Valid examples are:

| | |
|---|---|
| `[$x = 42`<br>`$x = $x + 1]` | `[$x = 42: $x = $x + 1]` |

### Text

Text not enclosed by square brackets is written directly to the output. To generate square brackets, escape them with a backslash: **`\[`** and **`\]`**. To generate a backslash, use **`\\`**.

### Comments

Comments inside an instruction block always begin with a `'` character, and terminate on the next line, or with a closing square bracket.

### Variables

Variables are created by assigning values to them. The **`$`** character is used when declaring or using a variable. Variable names are case-sensitive. Variables can be of the following types:

- Integer: Also used as boolean, where `0` is `false` and everything else is `true`
- String. *See also String Mechanisms* 1495.
- Object: Provided by MobileTogether. For example, the `$Options` 1497 object.

Variable types are declared by first assignment:

`[$x = 0]` means that x is now an integer.
`[$x = "teststring"]` means that x is now a string.

## Strings

Strings are enclosed in double quotes. **`\n`** and **`\t`** inside double quotes are interpreted as newline and tab, **`\"`** is a literal double quote, and `\\` is a backslash. String constants can also span multiple lines. String concatenation uses the **`&`** character:

```
[$BasePath = $outputpath & "/" & $JavaPackageDir]
```

## Objects

Objects are MobileTogether structures. Objects have properties, which can be accessed by using the **`.`** operator. It is not possible to create new objects in SPL, but it is possible to assign objects to variables. For example:

```
class [=$class.Name]
```

This example outputs the word `class`, followed by a space and the value of the `Name` property of the `$class` object.

## Conditions

Use **`IF`** statements, with or without the `ELSE` clause, as follows. Do not use round brackets around the condition.

```
if condition
        statements
else
        statements
endif
```

*Example*

```
[if $namespace.ContainsPublicClasses and $namespace.Prefix <> ""]
   whatever you want ['inserts whatever you want, in the resulting file]
[endif]
```

## Iterators

Use **`FOREACH`** statements to iterate, as follows:

```
foreach object in collection
           statements
next
```

*Example*

```
[foreach $class in $classes
   if not $class.IsInternal
]  class [=$class.Name];
[  endif
next]
```

## 23.3.2 String Mechanisms

SPL provides the string manipulation mechanisms that are listed below. These methods are applied on the input string itself.

`integer` **`Compare(s)`**
The return value indicates the lexicographic relation of the string to `s` (case-sensitive):

| | |
|---|---|
| `<0` | the string is less than s |
| `0` | the string is identical to s |
| `>0` | the string is greater than s |

`integer` **`CompareNoCase(s)`**
The return value indicates the lexicographic relation of the string to `s` (case-insensitive):

| | |
|---|---|
| `<0` | the string is less than s |
| `0` | the string is identical to s |
| `>0` | the string is greater than s |

`integer` **`Find(s)`**
Searches the string for the first match of a substring `s`. Returns the zero-based index of the first character of `s` or `-1` if s is not found.

`string` **`Left(n)`**
Returns the first `n` characters of the string.

`integer` **`Length()`**
Returns the length of the string.

`string` **`MakeUpper()`**
Returns a string converted to upper case.

`string` **`MakeUpper(n)`**

Returns a string, optionally with the first `n` characters converted to upper case.

`string` **`MakeLower()`**
Returns a string converted to lower case.

`string` **`MakeLower(n)`**
Returns a string, optionally with the first `n` characters converted to lower case.

`string` **`Mid(n)`**
Returns a string starting with the zero-based index position `n`.

`string` **`Mid(n,m)`**
Returns a string starting with the zero-based index position `n` and the length `m`.

`string` **`RemoveLeft(s)`**
Returns a string excluding the substring `s` if `Left(s.Length())` is equal to substring `s`.

`string` **`RemoveLeftNoCase(s)`**
Returns a string excluding the substring `s` if `Left(s.Length())` is equal to substring `s` (case-insensitive).

`string` **`RemoveRight(s)`**
Returns a string excluding the substring `s` if `Right(s.Length())` is equal to substring `s`.

`string` **`RemoveRightNoCase(s)`**
Returns a string excluding the substring `s` if `Right(s.Length())` is equal to substring `s` (case insensitive).

`string` **`Repeat(s,n)`**
Returns a string containing substring `s` repeated `n` times.

`string` **`Replace(sOld,sNew)`**
Replaces the string `sOld` with the string `sNew`.

`string` **`Right(n)`**
Returns the last `n` characters of the string.

`string` **`TrimLeft()`**
Returns the string after trimming spaces from the left. Trimming stops when a non-space character is encountered.

`string` **`TrimLeft(s)`**

Returns the string after removing, from the left, all the characters contained in s. Trimming stops when a character is encountered that is not contained in s.

**string TrimLeftRight()**
Returns the string after trimming spaces from the left and the right. Trimming stops when a non-space character is encountered.

**string TrimLeftRight(s)**
Returns the string after removing, from the left and the right, all the characters contained in s. Trimming stops at left and at right when a character is encountered that is not contained in s.

**string TrimRight()**
Returns the string after trimming spaces from the right. Trimming stops when a non-space character is encountered.

**string TrimRight(s)**
Returns the string after removing, from the right, all the characters contained in s. Trimming stops when a character is encountered that is not contained in s.

### String properties

The following properties are available:

- `Length`: returns the length of the string. *Example:* `$Options.deploymentPath.Length` returns the length of the string contained in `deploymentPath`.
- `XMLEncode`: returns the length of the string in XML-encoded format. *Example:* `$Options.deploymentPath.XMLEncode` returns the string contained in `deploymentPath` as XML-escaped text.

## 23.3.3 Properties of $Options

The **`$Options`** object can take the properties listed below. The values of most of these properties are usually provided in the Generate Program Code Wizard[1472], and are described there. The object's properties can be accessed by using the **.** operator. Some SPL template usage examples are given below.

```
<data
   android:scheme="[=$Options.schemeForRunSolutionUrl]"
   android:host="[=$Options.hostForRunSolutionUrl]"/>
```

```
@Override
public boolean GetServerUsesSsl()
   {
       return [if $Options.isUseSSL = 1]true[else]false[endif];
   }
```

## Workflow-related properties

The following workflow-related properties are available:

- **`workflowKey`**: returns the workflow key. *Example:* `$Options.workflowKey` returns the workflow key. Every time program code is generated[1472] and the associated workflow is deployed[1472] to the server, both are assigned the same unique workflow key. An appstore app will be able to access this workflow only if it has the same key as the workflow. See the final Deploy Design[1472] screen of the Generate Prgram Code[1472] process for details.
- **`deploymentPath`**: returns the path of the deployed workflow. *Example:* `$Options.deploymentPath` returns the workflow path on MobileTogether Server. Example of a workflow path: `/Public/DateTime/`.

## General properties

The values of these properties are provided in Screen 1 of the Generate Program Code Wizard[1472].

**`isTrialRunOnClientBuild`**
**`appName`**
**`visibleAppName`**
**`appVersion`**
**`hostForRunSolutionUrl`**
**`schemeForRunSolutionUrl`**

## User interface properties

The values of these properties are provided in Screen 2 of the Generate Program Code Wizard[1472].

**`splashScreenPortraitFilePath`**
**`splashScreenLandscapeFilePath`**
**`launcherIconFilePath`**
**`aboutLegal`**
**`aboutCopyRight`**

## Server properties

The values of several of these properties are provided in Screen 3 of the Generate Program Code Wizard[1472].

**`serverAddress`**
**`serverPort`**
**`isServerAccessAlwaysAnonymous`**
**`isUseSSL`**
**`isUsesZebraScanner`**
**`isUsesdatalogicScanner`**
**`isUsesZebraMobileCoputer`**

## Properties about user and app privileges

The values of these properties are provided in Screen 4 of the Generate Program Code Wizard[1472].

**`mayResetPersistentData`**
**`isAllowSMS`**

**isAllowTelephoneCall**
**isAllowCamera**
**isAllowLocationAccess**
**isAllowBluetooth**
**isAllowExternalStorageAccess**
**isAllowiCloudAccess**
**isAllowAudioRecording**
**isAllowPhotoGalleryAccess**

### Android and iOS properties

The values of these properties are provided in Screens 7 and 8 of the Generate Program Code Wizard[1472].

**targetDirectoryAndroid**
**androidPackageName**
**androidPackageDir**
**androidRoundLauncherIconFilePath**
**androidAdaptiveLauncherIconForegroundFilePath**
**isAndroidAdaptiveLauncherIconBackgroundColor**
**androidAdaptiveLauncherIconBackgroundFilePath**
**androidAdaptiveLauncherIconBackgroundColor**
**targetDirectoryIOS**
**iosBundleIdentifierPrefix**
**iosLauncherIconBackground**

### Windows App properties

The values of these properties are provided in Screens 9 and 10 of the Generate Program Code Wizard[1472].

**windowsAppCompanyName**
**windowsPhoneCompanyName**
**windowsCompanyPublisherID**
**targetDirectoryWindowsApp**
**windowsAppCompanyProductID**
**targetDirectoryWindowsPhone**
**windowsPhoneCompanyProductID**
**inputParameters** *(variable string in the form* `"par1=value1;par2=value2"`*)*

## 23.3.4 Properties of $Application

The **`$Application`** object can take the properties listed below. The object's properties can be accessed by using the **`.`** operator; see the *Examples* section below.

### Properties

**`CopyFile(sSource, sTarget)`**
Copies the file at the location `sSource` to the location `sTarget`.

**`UnzipFile(sZipFile, sTargetDir)`**
Unzips the ZIP archive at the location `sZipFile` to the directory `sTargetDir`.

**`FileExists(sFile)`**
Checks whether the file at location `sFile` exists. Returns **`1`** or **`0`**.

**`DeleteFile(sFile)`**
Deletes the file at the location `sFile`.

**`WriteDesignFileContent(sContentFile)`**
On code-generation, writes the content of the design file to the app-package using the file specified by `sContentFile`. (Deployed client files will also be written to the package.)

**`CopyAndResizeImage(sSource, sTarget, nTargetWidth, nTargetHeight)`**
Copies the image file at the location `sSource` to the location `sTarget`, and resizes the copied image to the pixel dimensions given by `nTargetWidth` and `nTargetHeight`.

### Examples

Here are some SPL template usage examples:

```
sub CopyFile( byval $sSource, byval $sTarget )
    return $Application.CopyFile( $sSource, $sTarget )
endsub

sub UnzipFile( byval $sZipFile, byval $sTargetDir )
    return $Application.UnzipFile( $sZipFile, $sTargetDir )
endsub
```

## 23.3.5 Miscellaneous Objects

The following objects can be accessed as variables:

| | |
|---|---|
| `$Host` | MobileTogetherDesigner <*version, e.g. 2.0*> |
| `$HostShort` | MobileTogetherDesigner |
| `$HostURL` | `http://www.altova.com/mobiletogether` |
| `$HostVersion` | Major product version (for example 1 when 1.5) |
| `$HostRelease` | Minor product version (for example 5 when 1.5) |
| `$RegisteredName` | Name of the licensed user |
| `$RegisteredCompany` | Licensed company |
| `$CreationDate` | Time of SPL code generation |

| | |
|---|---|
| `$outputpath` | Directory where the code is generated |

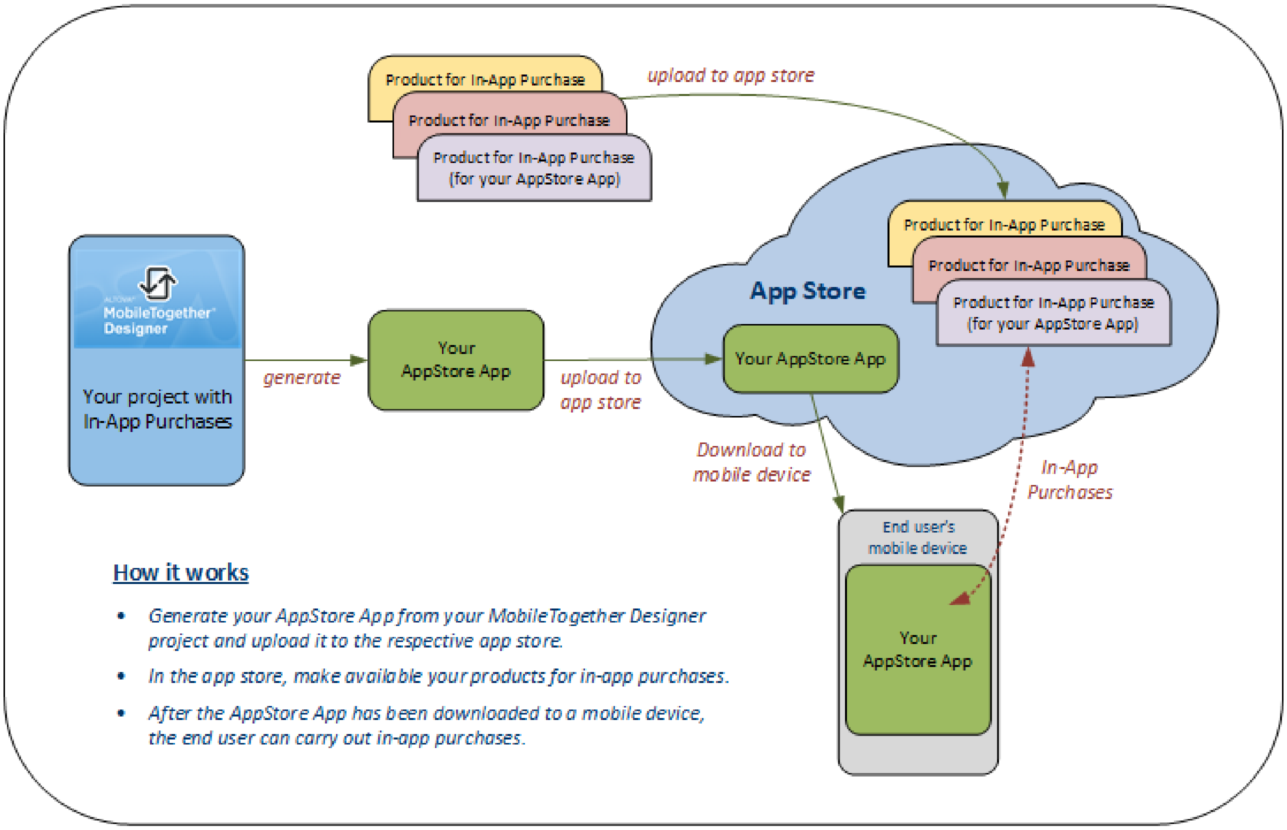
# 24 In-App Purchases

In-app purchases are content or subscriptions that you buy from inside an app on a mobile device. For example, you can buy *coins* from inside a game app or *a subscription* from inside a music app. The coins and the subscription are in-app purchases—because they have been purchased from inside an app.

MobileTogether Designer enables you to add in-app purchases as a feature of your AppStore Apps[1471].

In this section, we describe the mechanisms to add the in-app purchase functionality to the MobileTogether design of your AppStore App. After adding in-app purchases to your design, you would generate your AppStore App[1471] (from the design) for one or more of the supported platforms (Android, iOS, Windows). Note that the in-app purchase functionality only works in AppStore Apps[1471]; it will not work in MobileTogether solutions that run in MobileTogether Client—because of app-store restrictions.

You must of course make sure that the in-app purchases—the products that you want end users to buy—have been registered with the respective app store/s and are available for sale there. The end user of your AppStore App can then buy from the app store. In your MobileTogether design, you will design the interactions between your AppStore App[1471] and the app store so as to correctly transact an in-app purchase.


Product for In-App Purchase
Product for In-App Purchase
Product for In-App Purchase (for your AppStore App)
upload to app store
MobileTogether Designer
Your project with In-App Purchases
generate
Your AppStore App
upload to app store
App Store
Your AppStore App
Product for In-App Purchase
Product for In-App Purchase
Product for In-App Purchase (for your AppStore App)
Download to mobile device
In-App Purchases
End user's mobile device
Your AppStore App

How it works

- Generate your AppStore App from your MobileTogether Designer project and upload it to the respective app store.
- In the app store, make available your products for in-app purchases.
- After the AppStore App has been downloaded to a mobile device, the end user can carry out in-app purchases.

Below, we first provide an overview of the categories of in-app purchases and then outline the broad steps for setting up in-app purchases in a MobileTogether design. In the sub-sections of this section, we describe these steps in detail. The section is rounded off with a description of an example project[1517], which is delivered with your installation of MobileTogether Designer and located in the Tutorials folder of MobileTogether Designer.

*App store documentation*

For information about the procedures at the different app stores, see the relevant documentation at the respective app store. It is important that you understand the workflow requirements for each app store and make sure that you implement them correctly when creating the design of your AppStore App 1471.

The links below provide entry points to the app store documentation.

- Android/Google: https://developer.android.com/google/play/billing
- iOS/Apple: https://developer.apple.com/documentation/storekit/in-app_purchase
- Windows/Microsoft: https://docs.microsoft.com/en-us/windows/uwp/monetize/
- Windows/Microsoft: https://docs.microsoft.com/en-us/windows/uwp/publish/set-app-pricing-and-availability
- Windows/Microsoft: https://docs.microsoft.com/en-us/windows/uwp/publish/add-on-submissions

## Categories of in-app purchases

There are three categories of in-app purchases:

- *Consumables:* These are purchases that are consumed within the app, such as game currency. They are bought once, depleted, and can be purchased again.
- *Non-consumables:* These are content or services that are purchased once and do not expire, for example, filters in a photo app.
- *Subscriptions:* These are access to periodically updated content or services, such as a monthly magazine. Subscription charges recur until canceled.

## Setting up in-app purchases for an MT project

The broad steps for enabling in-app purchases in your AppStore App are as follows:

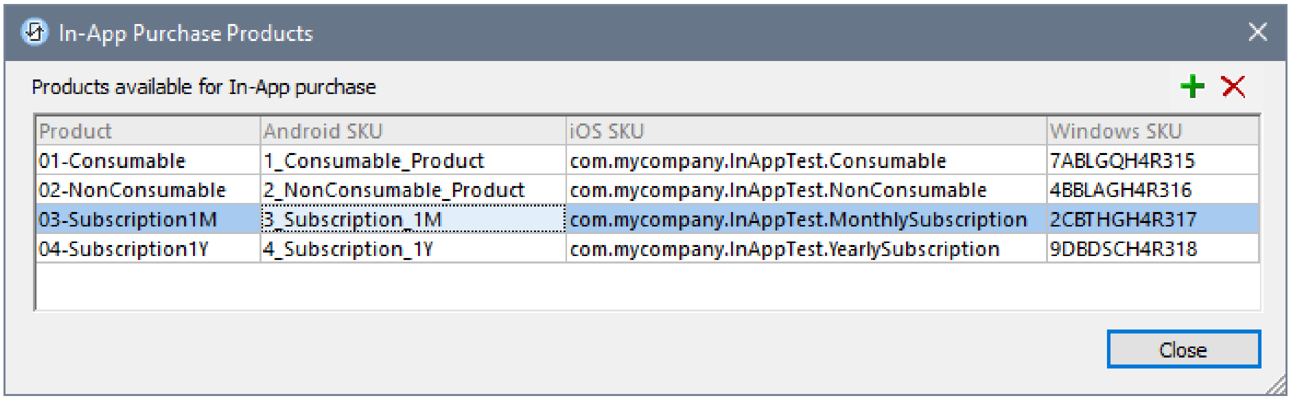
1. Via your app store account, create the in-app products (or SKUs, stock-keeping units) that you want to make available as in-app purchases. During this process, you will give each product, among other properties, an ID, a name and description, and set the product's price. Product IDs in the different app stores each have a specific and different format. So note down these product IDs; you will need them in the MT design when you map the store IDs to the product's name in the design. In the design, you will work with this single product name.
2. In the MT design, for each product that you want to make available as an in-app purchase, map that product's IDs in the different app stores to one product name 1505. This product name will be used in the design as the product's identifier. By using a common name for a product's different IDs in the respective app stores, we obviate referencing, in the design, a single product via multiple identifiers.
3. Information about *products* is stored at the app store. This information can be retrieved by MobileTogether's Query Available Products 935 action, and will then also be stored in the design's In-App-Purchase Page Source 1507.
4. Information about the current user's *purchases* is also stored at the app store. It can be retrieved by MobileTogether actions and is also stored in the design's In-App-Purchase Page Source 1507.
5. The product and purchase information stored in the In-App-Purchase Page Source 1507 can be displayed in the design by using suitable controls 403.
6. To make an in-app purchase, the end user triggers the Purchase action 933, which identifies the product to be purchased and the current user.
7. The app store executes the purchase transaction 1511 and sends transaction data to the app on the mobile device.
8. In the app, the purchased product (for example, a subscription) is activated (by you, the designer, through some other mechanism related to the product).
9. The app acknowledges the purchase 1511, and this completes the transaction.

## 24.1 Register Products

The products that you want to sell in your app must be:

- registered and available in the respective app store (Google, Apple, Windows), and
- registered in the MobileTogether design under a single product name; essentially, the respective IDs of the product in the different app stores are mapped to a single name that serves as the product's ID in the design (*see the screenshot below*).

During the registration (or creation) of the product, you will have given the product an ID and set its price (among other things). Since each app store specifies a different format for the ID of products (compare the SKU ID formats in the screenshot below), the different IDs are mapped to a single product name (in the *Product* column of the dialog shown below). In the MT design, a product is identified by the name given to it in this dialog. When the respective AppStore Apps[1471] are generated from the design, the appropriate SKU ID is used for each platform. This mapping mechanism enables all *design procedures* related to a single product to be handled in a unified way in the design regardless of platform—by using the product's name, as defined here.



| Product | Android SKU | iOS SKU | Windows SKU |
|---|---|---|---|
| 01-Consumable | 1_Consumable_Product | com.mycompany.InAppTest.Consumable | 7ABLGQH4R315 |
| 02-NonConsumable | 2_NonConsumable_Product | com.mycompany.InAppTest.NonConsumable | 4BBLAGH4R316 |
| 03-Subscription1M | 3_Subscription_1M | com.mycompany.InAppTest.MonthlySubscription | 2CBTHGH4R317 |
| 04-Subscription1Y | 4_Subscription_1Y | com.mycompany.InAppTest.YearlySubscription | 9DBDSCH4R318 |

To set up the ID-to-Product-name mapping, do the following:

1. In MobileTogether Designer, select **Project | In-App Purchase Products**.
2. In the In-App Purchase Products dialog that appears (*screenshot above*), enter a name for each product that you want to make available for purchase.
3. In the respective SKU columns, enter the respective product IDs that you assigned to the product when the product was registered at the respective app store.

Click the **Plus** icon to add a new product. Click the **Delete** symbol to remove a product. After you have entered the data for all the products you want to make available for in-app purchases, click **Close**.

Also see how to create an in-app product for Android and how to create a subscription at the Google Play Store.

### XPath functions for retrieving product name and ID

MobileTogether Designer provides two functions related to the naming mechanism: for retrieving (on a given platform) the product name from the product ID, and vice versa. You can use these functions at any location and at any time in the workflow.

*Retrieve platform-specific product ID from product name*

The `mt-in-app-purchase-product-to-platform()` function takes as its single argument a product name as entered in the In-App Purchase Products dialog (*see screenshot above*). It returns the ID of the product on the current platform. In simulations, the current platform is the platform of the simulation device. For example,

> **`mt-in-app-purchase-product-to-platform("03-Subscription1M")`** returns **`"3_Subscription_1M"`** on Android devices and **`"com.mycompany.InAppTest.MonthlySubscription"`** on iOS devices.

*Retrieve product name from platform-specific product ID*

The `mt-in-app-purchase-platform-to-product()` function takes as its single argument a product ID as defined for the current platform in the In-App Purchase Products dialog (*see screenshot above*). It returns the product name of the submitted product ID. In simulations, the current platform is the platform of the simulation device. For example,

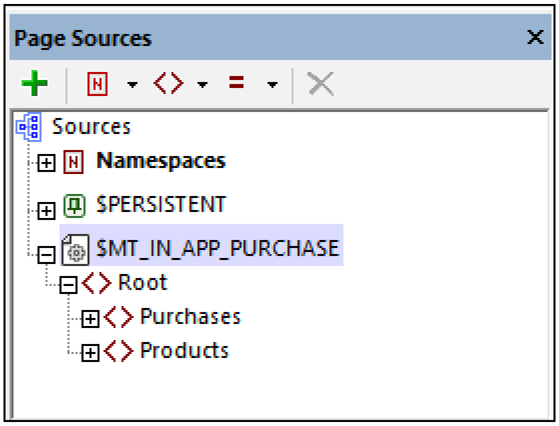
> **`mt-in-app-purchase-platform-to-product("3_Subscription_1M")`** returns **`"03-Subscription1M"`** on Android devices.

See MobileTogether Extension Functions 1262 for a complete listing and description of MT functions.

## 24.2 In-App-Purchase Page Source

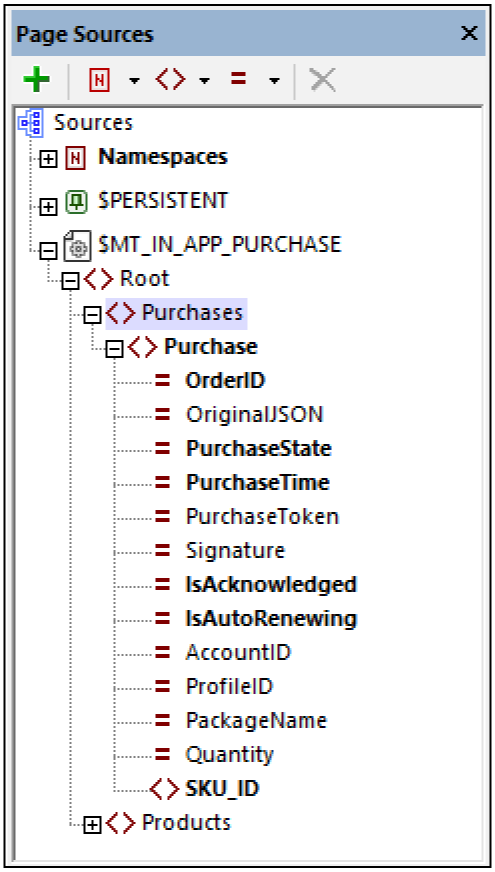
Data about your products that are available for in-app purchases are stored in the respective app stores. Data about purchases made by an end user are also stored in the respective app stores.

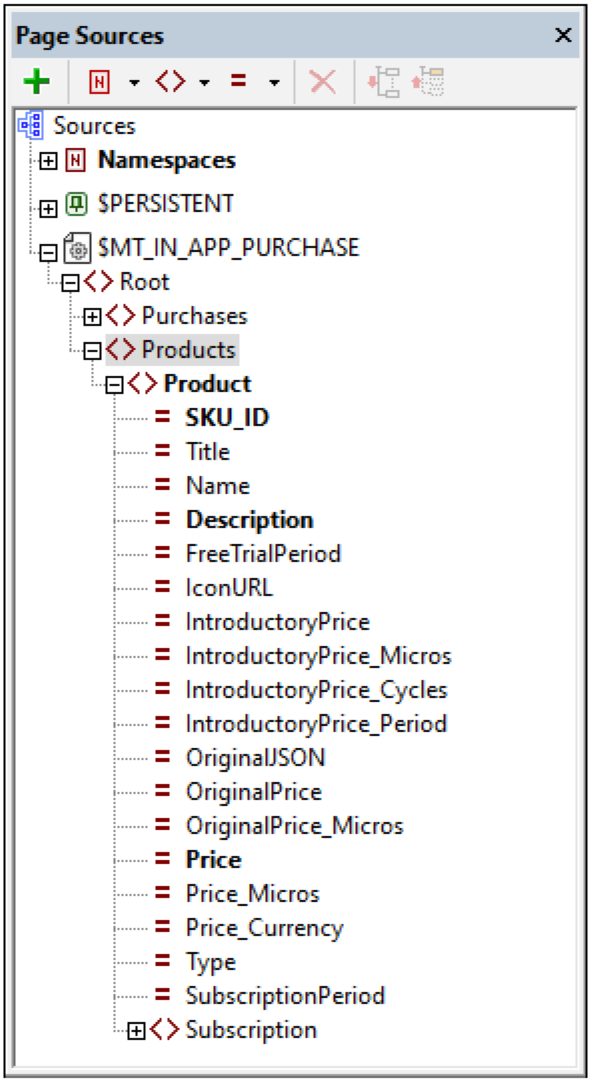
When this data is needed on the client device for an in-app purchase, it is requested by the client from the app store and downloaded to a special page source in your MT design: `$MT_IN_APP_PURCHASE` (*see screenshot below*). During the design phase, this page source is added automatically to the design the first time you add any of the In-App Purchase actions 931.



The **`Root`** element of the page source contains two elements:

- **`Purchases`** contains data about individual purchases, each of which is stored in a child **`Purchase`** element (*see tree in screenshot below*). The attributes of the **`Purchase`** element correspond to the data points of a purchase at the app stores. While the actual data points are different (in number and name) from store to store, they are all mapped to the attributes of the **`Purchase`** element. This enables the design to reference a common set of abstract data points that can be used across all platforms. Here is an example of how the **`Purchase`** element is used: The Query Purchases 934 action gets data about one or more purchases and places the data of each purchase in a separate **`Purchase`** element *(see screenshot below left)*.
- **`Products`** contains data about products that are available for in-app purchases. Data pertaining to each product is stored in a separate child **`Product`** element. Just as with purchases, while the actual data items relating to a product are different from store to store, this data is stored in a single set of attributes of the **`Product`** element. Here is an example of how the **`Product`** element is used: The Query Available Products 935 action returns data about the queried product/s and places the data of each product in a separate **`Product`** element *(see screenshot below right)*.





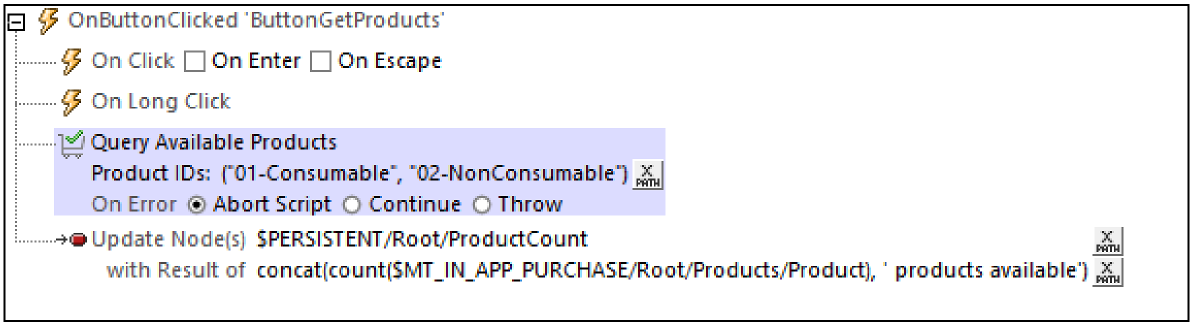
Note the following points:

- The nodes of the tree constitute a superset that is a union of the data points used by the different app stores. Not all nodes will be used by a single app store. For example, the **`Subscription`** element is used only by the Google Play Store (Android devices).
- While `SKU_ID` is an attribute of the `Product` element, it is a child element of the `Purchase` element.
- Purchase data from an app store is typically sent as a JSON string. MobileTogether retrieves key data points from this purchase data and saves them, for each purchase, in the attributes of a `Purchase` element. On Android and Windows systems, the original JSON string containing the data is stored in an `@OriginalJSON` attribute of the **`Purchase`** element, whereas on iOS systems the original JSON string is stored as an attribute of the **`Purchases`** element. This is because the Apple Store, when queried for the purchase data of an end user, returns data of all purchases rather than the data of each purchase separately (see the description of the Restore Purchases 934 action). For a description of how to query purchases from the Google and Windows app stores, see the topic Query Purchases 934.

## 24.3 Query Available Products

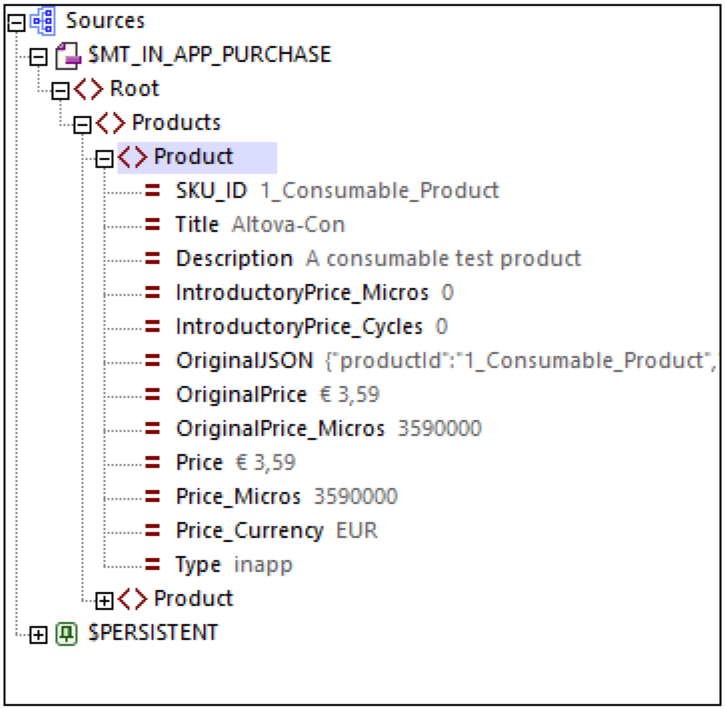
In order for an end user to be able to buy a product, the products that are available for in-app purchase must be listed in the app. The mechanism to obtain a list of available products works as follows:

1. The Query Available Products[935] action queries the app store for the details of a set of products. These products are identified in the design by their names, which are submitted as a sequence of strings (*see screenshot below*). If no product is submitted, then all the products defined in the In-App Purchase Products [1505] dialog are queried.



```
⊟ OnButtonClicked 'ButtonGetProducts'
    On Click □ On Enter □ On Escape
    On Long Click
    Query Available Products
    Product IDs: ("01-Consumable", "02-NonConsumable")
    On Error ● Abort Script ○ Continue ○ Throw
    Update Node(s) $PERSISTENT/Root/ProductCount
      with Result of concat(count($MT_IN_APP_PURCHASE/Root/Products/Product), ' products available')
```

2. The app store returns the data about the queried products, and this data is stored in the `$MT_IN-APP_PURCHASE` page source. Each product's details is stored in a **`Product`** child element of **`Products`** (*see screenshot below*).



```
⊟ Sources
  ⊟ $MT_IN_APP_PURCHASE
    ⊟ Root
      ⊟ Products
        ⊟ Product
            = SKU_ID 1_Consumable_Product
            = Title Altova-Con
            = Description A consumable test product
            = IntroductoryPrice_Micros 0
            = IntroductoryPrice_Cycles 0
            = OriginalJSON {"productId":"1_Consumable_Product",
            = OriginalPrice € 3,59
            = OriginalPrice_Micros 3590000
            = Price € 3,59
            = Price_Micros 3590000
            = Price_Currency EUR
            = Type inapp
        ⊞ Product
  ⊞ $PERSISTENT
```

3. Since the product data is now in a page source, any data from the page source tree can be displayed in the design. A suitable way to display all available products would be via a dynamic table[1069], with each product being displayed in a row group of the table. You do not need to display all details of a product, but can select only those data points that you want to display: for example, the product

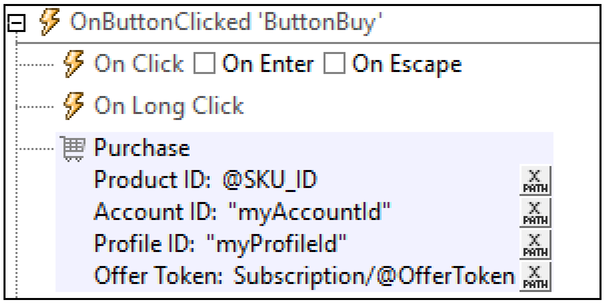
name, its description, and its price. Run a simulation of the example project 1517 to see how the mechanism to query available products works.

**Note:** In the Query Available Products 935 action, if no specific product is specified, then all the products defined in the In-App Purchase Products 1505 dialog are queried.

## 24.4 Purchase Products

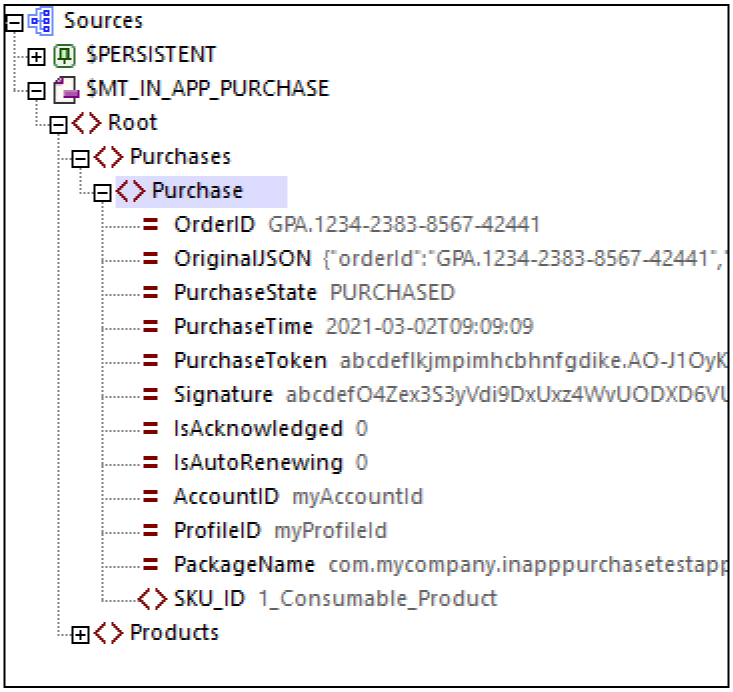
The core procedure of an in-app process is the transaction that carries out the actual purchase. It is implemented via the Purchase action[933], and it works as follows:

1. A Purchase action[933] sends a purchase request from the app to the app store. The request contains (i) the name of the product to be purchased, (ii) details of the user account to be used for the transaction, and (iii) the offer token that is required for Android store subscriptions (*see screenshot below*).



OnButtonClicked 'ButtonBuy'
On Click □ On Enter □ On Escape
On Long Click
Purchase
Product ID: @SKU_ID
Account ID: "myAccountId"
Profile ID: "myProfileId"
Offer Token: Subscription/@OfferToken

2. The app store responds with a summary of the product's key data and asks for confirmation of the purchase. The end user on the client device confirms the purchase.
3. The app store carries out the transaction and sends data about the transaction to the app. This data about the purchase is stored in the `$MT_IN_APP_PURCHASE` page source, inside a **`Purchase`** element (*see screenshot below*).



```
Sources
  $PERSISTENT
  $MT_IN_APP_PURCHASE
    Root
      Purchases
        Purchase
          OrderID  GPA.1234-2383-8567-42441
          OriginalJSON  {"orderId":"GPA.1234-2383-8567-42441",
          PurchaseState  PURCHASED
          PurchaseTime  2021-03-02T09:09:09
          PurchaseToken  abcdeflkjmpimhcbhnfgdike.AO-J1OyK
          Signature  abcdefO4Zex3S3yVdi9DxUxz4WvUODXD6VU
          IsAcknowledged  0
          IsAutoRenewing  0
          AccountID  myAccountId
          ProfileID  myProfileId
          PackageName  com.mycompany.inapppurchasetestapp
          SKU_ID  1_Consumable_Product
      Products
```

4. Once the purchase data has been received in the design, the actions of the **`OnPurchaseUpdated`**[296] event are triggered. These are actions that you define. Typically, you would carry out the following steps:

i. Check the response code[1513] that the app store sends about the success/failure of the purchase request. You can retrieve the response code by using the function `mt-last-in-app-purchase-response-code()`. If the purchase request was successful, the function will return `0`. Other return codes vary across platforms. (See the topic Other Operations[1513] for additional diagnostic functions, such as to check whether the user canceled the purchase.)
ii. If needed, save the purchase state of the product outside the `$MT_IN_APP_PURCHASE` page source (either in **`$PERSISTENT`**, some other page source, or an external file). You might want to do this if you want to keep, for some particular reason, a separate history of purchase activity.
iii. Activate the purchased product in the app. After checking the purchase state, subscription validity, possible cancellation, etc, of the purchase, you will need to activate the product accordingly.
iv. Acknowledge the purchase (via the Acknowledge Purchase[936] action). You, as the designer, must trigger the Acknowledge Purchase[936] action in order to finalize the purchase so that the vendor of the app (which could be you or some other entity) receives payment. Note that acknowledgment may be one of two types: (i) for subscriptions and non-consumable products, and (ii) for consumable products. For details, see the Acknowledge Purchase[936] action and Example Project: The OnPurchaseUpdated Event[1529]. You should acknowledge the purchase after you have activated the product (see previous step); this would ensure that the end user receives the use of the purchase before any data related to the purchase is possibly modified.

Once a purchase has been acknowledged, the in-app purchase will be complete. Afterwards, you can query purchases and display information about purchases by using the Query Purchases[934] or Restore Purchases[934] actions. In the case of consumables that have been purchased for Windows devices, you can also use the Get/Report Consumable[937] action to query how many credit units are still available (balance) and how many have been consumed (fulfilled).

## 24.5 Other Operations

The preceding topics of this section described the main components of MobileTogether Designer's In-App Purchases mechanism. In this topic, we list and briefly describe the other key components of the mechanism.

### Check availability of In-app Purchases service

You can use a function named **`mt-in-app-purchase-service-started()`** to check whether the in-app purchase service has been started on the client device. For details, see the topics MobileTogether Extension Functions 1262 and Example Project: Availability of In-App Service 1521.

### Check app store response to last in-app purchase

The following MobileTogether extension functions 1262 can be used to check the success of the last request to the app store that was related to an in-app purchase:

- `mt-last-in-app-purchase-response-code()`
- `mt-last-in-app-purchase-response-text()`
- `mt-last-in-app-purchase-response-was-user-canceled()`

This enables you to make the next step in the workflow dependent on the success of the current step. For an example, see the actions of the The OnPurchaseUpdated Event 1529 in the Example Project 1517. Note that these functions can be used to obtain the response of the app store to the last request made to it that was related to an in-app purchase.

An app store might respond with any of a number of responses, depending on which app store is involved. A code of `0` indicates a successful operation. The other codes indicate different types of failure. The function that returns response text would provide more information about the failure. One type of failure, however, should be handled differently since it is not an error. This would be when the end user cancels a purchase. To distinguish this type of purchase failure from errors, MobileTogether provides the function `mt-last-in-app-purchase-response-was-user-canceled`.

### OnPurchaseUpdated: when an app store accepts a purchase request

When an app store accepts a purchase request, the purchase is automatically added to the In-App-Purchase Page Source 1507. The purchase is said to be updated (in the page source) and the **`OnPurchaseUpdated`** event is triggered. A sequence of actions can be defined for the event, which enables you to seamlessly carry out actions, without requiring any user input, after a purchase request has been accepted. For example, you might want to acknowledge a purchase directly after it has been accepted (*see next topic below*). To access the event's Actions dialog, go to the project property In-App Purchase Actions 296.

Note the following points about the actions of the **`OnPurchaseUpdated`** event:

- While the actions of the **`OnPurchaseUpdated`** event are being processed, the `$MT_UpdatedInAppPurchases` variable is in scope. The variable contains a sequence of strings, which are the SKU_IDs of purchased products that have been updated in the In-App-Purchase Page Source 1507. So, if one item has been purchased, then its SKU_ID can be obtained with the XPath expression **`$MT_UpdatedInAppPurchases[1]`**. The variable goes out of scope once the event's actions have been processed. This means that the variable can only be used in actions of the **`OnPurchaseUpdated`** event.

- In the In-App-Purchase Page Source[1507], each **`Purchase`** element has a `PurchaseState` attribute that takes a value of either `PENDING` or `PURCHASED`. You might want to suitably modify your workflow to take this status into account.

For an example (i) of the kind of actions that can be defined for the `OnPurchaseUpdated` event, and (ii) of how to use the `$MT_UpdatedInAppPurchases` variable, see the description of this event in the example project: The OnPurchaseUpdated Event[1529].

## Activate product

Activate the purchased product in the app. After checking the purchase state, subscription validity, possible cancellation, etc, of the purchase, you will need to activate the product accordingly. This is a step that you as the designer of the app must carry out; MobileTogether cannot automatically do this for you.

## Acknowledge Purchase

After the end user successfully buys a product via an in-app purchase, the app store accepts the request and sends information about the purchase to the app. On receiving this information, the purchase must be **`acknowledged`** from the app side, and this is something that you, as the app designer, must do silently without bringing the end user into the picture. The acknowledgment step enables the app to process the information received by the app store to determine whether the purchase is legitimate and whether to close the purchase from the vendor side. This provides you, as the app designer, with an opportunity to verify different aspects of the purchase. The app stores provide documentation that give you detailed information about what you should assess before acknowledging the purchase. Kindly read the relevant app store information before deciding on the steps to take before acknowledging. It is only after you acknowledge the purchase, that the app store will process the payment and route it to the product-owner's account.

Purchases are acknowledged via MobileTogether's Acknowledge Purchase[936] action, and it is best to include this action as one of those to be carried out when the **`OnPurchaseUpdated`** event is triggered (*see previous section above*).

For details of this action, see the description of the Acknowledge Purchase action[936] and the description of its implementation in the example project[1529].

## Query stores for purchase data

You can query the app store for all the purchases transacted by the current user. The action to do this is the Query Purchases action[934]. When the action is executed, the app store returns data about the user's purchases, and key datapoints are stored in the **`Purchases`** element of the In-App-Purchase Page Source[1507]. Each purchase is stored in an individual `Purchase` element. You can then use this purchase data in the app's workflow. For detailed information, see the description of the Query Purchases action[934] and the description of the Query Purchases button in the example project[1530].

## Restore purchases (iOS only)

The Apple Store keeps a record of all of an end user's purchases and enables the end user to restore all these purchases on an iOS device at any time. The Restore Purchases[934] action can be used to query the Apple Store for all of an end user's purchases. The data returned is stored in the **`Purchases`** element of the In-App-Purchase Page Source[1507], with each purchase being stored in an individual `Purchase` element. You can then use this purchase data in the app's workflow. For detailed information, see the description of the Restore Purchases[934] action and the description of the Restore Purchases button in the example project[1530].

## Get/Report Consumables (Windows only)

After a consumable[1502] has been purchased, it can be consumed within the app. In the case of Windows apps, the Windows app store offers an option to keep a record of a user's current consumable balance. The Get/Report Consumable action[937] addresses this Windows option. It can get the current balance from the Windows app store and report the fulfillment level of the consumable to the Windows app store. For details see the description of the action[937].

**Note:** It is your responsibility to check the current level of consumable credit, especially in the case of iOS and Android, where the respective app stores—unlike the Windows app store—does not offer an option for tracking consumable usage.

## Keep track of the product's activation status

It is your responsibility to determine whether a product is active within the app at any given time. This decision would be based on the following factors:

- The product's *PurchaseState* as reported by the app store.
- In the case of consumable products, the current level of consumable credit. You will have to keep track of this in your app.
- In the case of subscriptions, the current state of the subscription relative to the start date and period of the subscription.
- Any other factor that affects the activation status of the product, such as a free extension period for a subscription.

## REST interfaces

You can also use REST interfaces as far as these are supported on the respective platforms. The use of REST interfaces with AppStore Apps on Android has been tested, and the documentation of these interfaces is available here: https://developers.google.com/android-publisher/api-ref/rest.

# 24.6 Simulation and Testing

You can test the in-app purchases of your AppStore App by simulating it in two easy ways, as described below.

## Simulation in the Designer

This simulation enables you to test the workflow with product and purchase data that is stored in an XML file.

This XML file is located by default at **`"<Program Files>\Altova\MobileTogetherDesigner7\InAppPurchase\InAppPurchase Samples.xml"`**. Make sure that you map the SKU IDs in this file to a product name as described in the topic Register Products 1505. Note that you can save your XML data file at any other location if you like; in this case, change the reference to the file in the Simulation 2 tab of the Options dialog 1669.

Note that the simulation in MobileTogether Designer works with static product and purchase data, which is taken from the XML file listed above. While a designer simulation broadly simulates the actual procedure, a trial run on client simulation (*see below*) more closely simulates the various steps of the live procedure.

## Simulating trial run on client

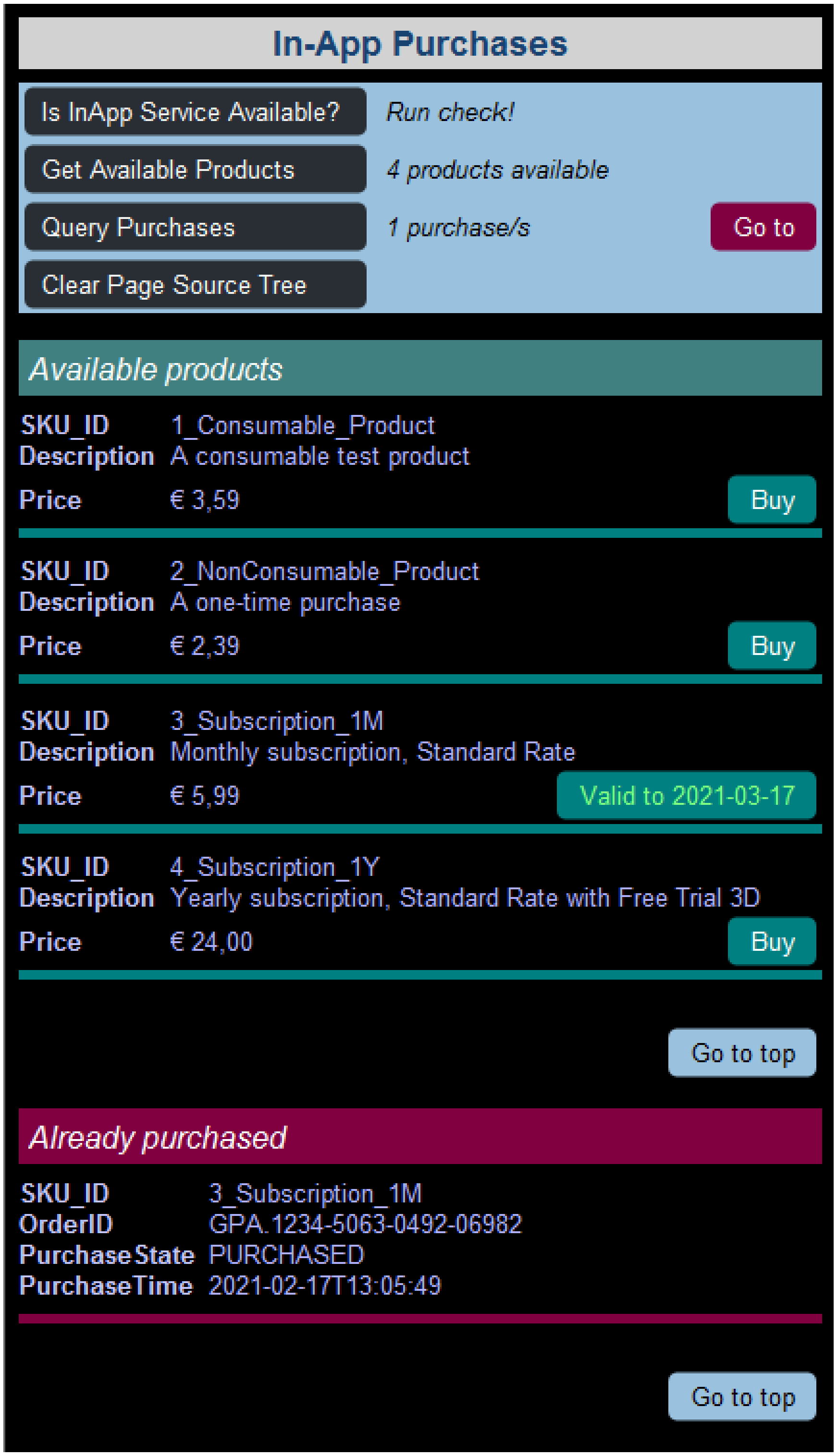
While a simulation in the designer that uses dummy data is useful for testing the workflow steps on a broad level, a significant problem with Designer simulations is that you cannot use real app store data for the simulation. A better way to test your AppStore App 1471 is to generate a Trial Run build of the app that can directly use the design that you are creating in MobileTogether Designer. To generate a Trial Run build, make sure that the following two settings are correctly selected when you build the app 1472:

- In Screen 1 1472, in the *Build Modes* pane, select *Trial Run on Client*.
- In Screen 3 1472, enter the correct *IP Address* and *Port* settings so that the app can connect to your MobileTogether Designer.

Once the Trial Run build has been generated, it will connect to the design in MobileTogether Designer and use the latest version of the design. This saves you the trouble of having to regenerate the AppStore App to test it each time you make a change to the design. Once you are satisfied with your tests and ready to publish, you can generate a final publication build that you can upload to the app store.

## 24.7 Example Project

An example project named `InAppPurchases.mtd` is located in your *(My) Documents* folder `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\Tutorials\InAppPurchases`. The project implements a simple in-app purchase workflow to demonstrate the MobileTogether mechanism for in-app purchases. For you to get a broad understanding of how to set up in-app purchases from your project, we recommend that you run a simulation of the project in MobileTogether Designer 1356 and then inspect the settings of various project components. The topics of this section describe the key components of the example project.



Run check!

4 products available

1 purchase/s

| | |
|---|---|
| SKU_ID | 1_Consumable_Product |
| Description | A consumable test product |
| Price | € 3,59 |

| | |
|---|---|
| SKU_ID | 2_NonConsumable_Product |
| Description | A one-time purchase |
| Price | € 2,39 |

| | |
|---|---|
| SKU_ID | 3_Subscription_1M |
| Description | Monthly subscription, Standard Rate |
| Price | € 5,99 |

Valid to 2021-03-17

| | |
|---|---|
| SKU_ID | 4_Subscription_1Y |
| Description | Yearly subscription, Standard Rate with Free Trial 3D |
| Price | € 24,00 |

| | |
|---|---|
| SKU_ID | 3_Subscription_1M |
| OrderID | GPA.1234-5063-0492-06982 |
| PurchaseState | PURCHASED |
| PurchaseTime | 2021-02-17T13:05:49 |

Since the simulation in MobileTogether Designer does not use a real account or connect to an app store, it uses data in the file `InAppPurchase Samples.xml` to simulate products and purchases. This data file is located in your Program Files folder `Altova\MobileTogetherDesigner10.1\InAppPurchase`. It has already been set as the default file to use for the simulation of in-app purchases[1669], in the Simulation 2 tab[1669] of the Options dialog (**Tools | Options**[1663]).

**Note:** Since the simulator uses the device selected in the *Preview Device*[254] combo box, it will simulate purchases at the corresponding app store. To simulate purchases at a different app store, change the preview device suitably.

### The data in the file "InAppPurchases Samples.xml"

The broad structure of the data file is as follows:

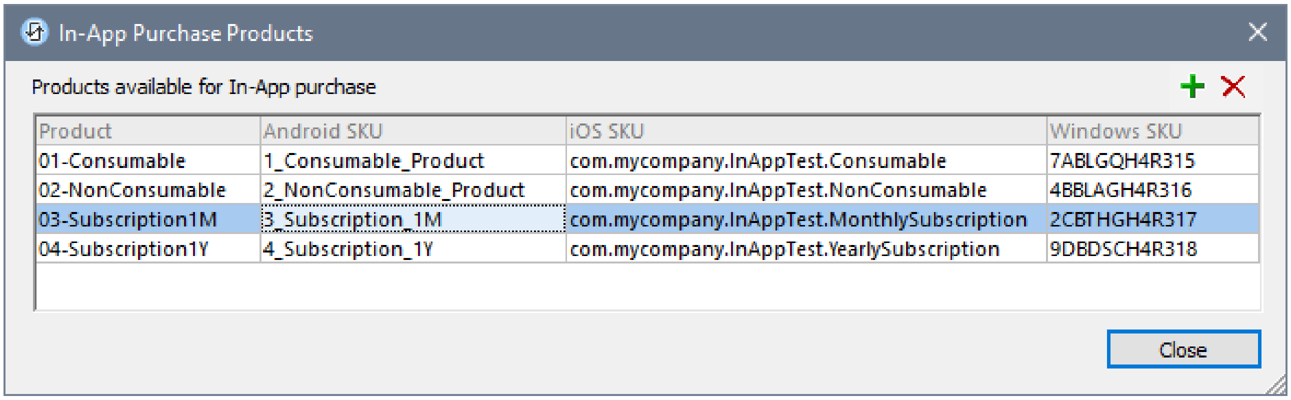
```
Root
|---Android
|   |---Products
|   |   |---Product
|   |   |   |---Subscription
|   |---Purchases
|   |   |---Purchase
|---iOS
|   |---Products
|   |   |---Product
|   |---Purchases
|   |   |---Purchase
|---Windows
|   |---Products
|   |   |---Product
|   |---Purchases
|   |   |---Purchase
```

Note one major difference between the data structure for Android devices on the one hand and those of iOS and Windows devices on the other. On Android, each subscription plan is considered to be a variant of a product and is identified by data in a corresponding **`Subscription`** element of the product. On iOS and Windows devices, however, each subscription plan is considered to be a separate product. The data structure on these latter devices therefore do not contain the **`Subscription`** element.

The data file contains four products and four purchases for each platform/store. Note that the data in the example file will be selected according to which device is selected for the simulation in the *Preview Device*[254] combo box. For example, if an Android device has been selected as the *Preview Device*[254], then data inside the **`Android`** element will be used for the simulation.

## 24.7.1 Map Product IDs to Product Name

Each product will have a different SKU ID at each app store. In the In-App Purchase Products dialog[1505], for each product, map the three SKU IDs to a single product name. This name will be used to identify the product in the design.



In-App Purchase Products

Products available for In-App purchase

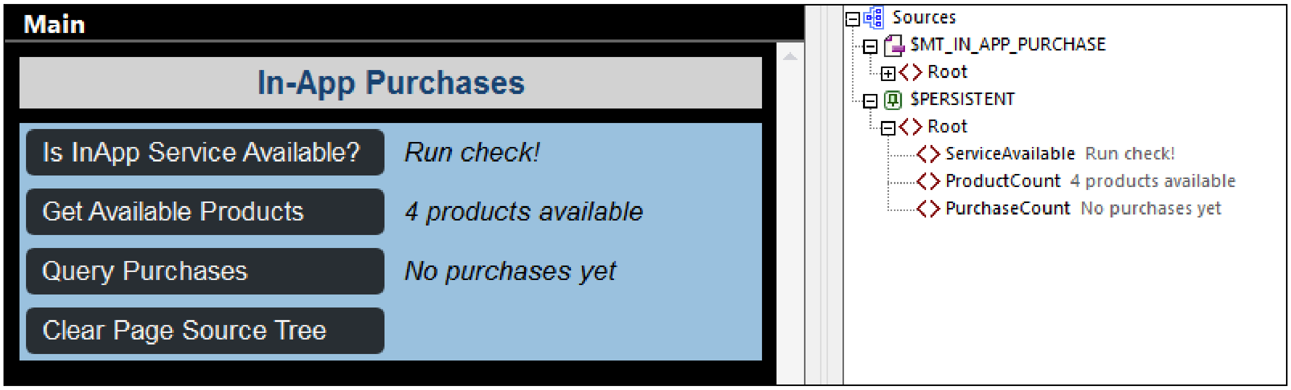
| Product | Android SKU | iOS SKU | Windows SKU |
|---|---|---|---|
| 01-Consumable | 1_Consumable_Product | com.mycompany.InAppTest.Consumable | 7ABLGQH4R315 |
| 02-NonConsumable | 2_NonConsumable_Product | com.mycompany.InAppTest.NonConsumable | 4BBLAGH4R316 |
| 03-Subscription1M | 3_Subscription_1M | com.mycompany.InAppTest.MonthlySubscription | 2CBTHGH4R317 |
| 04-Subscription1Y | 4_Subscription_1Y | com.mycompany.InAppTest.YearlySubscription | 9DBDSCH4R318 |

The SKU IDs are taken from the data file you use in simulations (*see above*).

## 24.7.2 The $PERSISTENT Page Source

The **`$PERSISTENT`** page source has three elements (*see pane at right in screenshot below*).

- `ServiceAvailable`: which holds text that indicates whether the In-App Purchases service is available 1521 on the device
- `ProductCount`: which holds the count of available products
- `PurchaseCount`: which holds the count of purchases

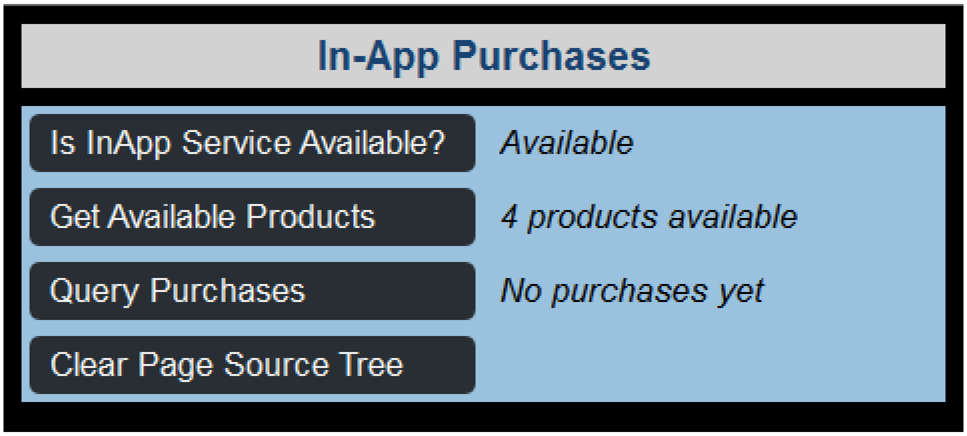


The values of these elements are each displayed in a label, done by setting that element as the page source link of the label. Each label is located to the right of a button in the blue table (*see screenshot*). When a button is clicked, the corresponding node in the **`$PERSISTENT`** page source is updated, and the updated node value is displayed in the label corresponding to that button.

## 24.7.3 Availability of In-App Service

Each operating system (Android, iOS, Windows) offers a service that sets up the device for in-app purchases at the corresponding app store. The MobileTogether function `mt-in-app-purchase-service-started()` checks whether the service has been started on the respective operating system. It returns **`true()`** or **`false()`**.

In the example project, this functionality has been enabled as a button-click action on the button *Is InApp Service Available?* (*see screenshot below*).

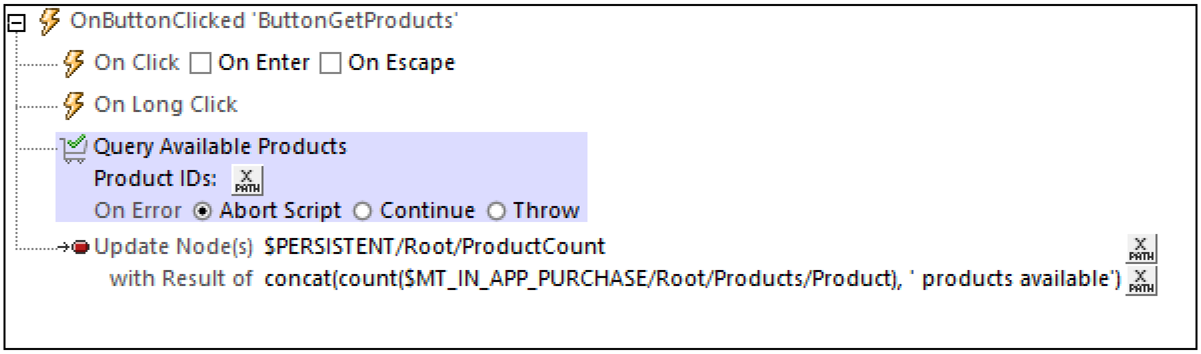


When the button is clicked, the function `mt-in-app-purchase-service-started()` is called and the **`ServiceAvailable`** node in the **`$PERSISTENT`** tree is updated based on the value returned by the function. Since this node is the page source link of the label on the right-hand side of the button, the node's value is immediately displayed in the label.

## 24.7.4 Get Available Products

On clicking the button *Get Available Products*, the Query Available Products 935 action is executed. At run time, the action would get the products from the app store. In our simulation, the action gets the products from the simulation data file 1517.

In our example project, the action is defined with the *Product IDs* setting empty (*see screenshot below*). As a result, all the products defined in the In-App Purchase Products 1505 dialog are queried.

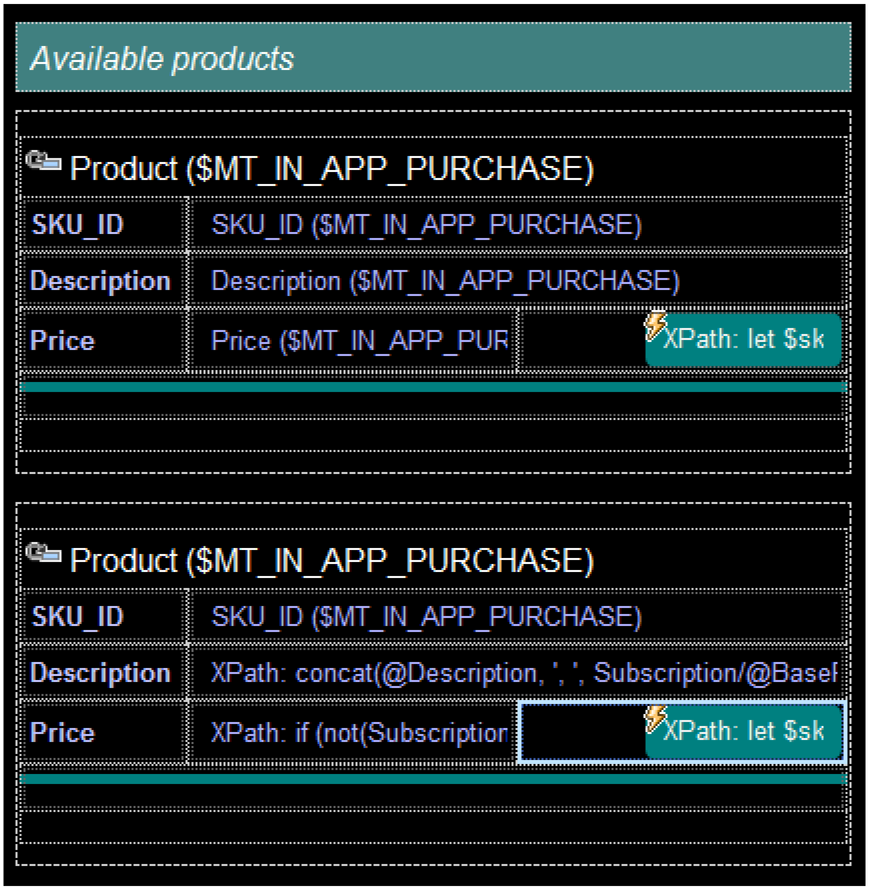


OnButtonClicked 'ButtonGetProducts'
On Click □ On Enter □ On Escape
On Long Click
Query Available Products
Product IDs:
On Error ● Abort Script ○ Continue ○ Throw
Update Node(s) $PERSISTENT/Root/ProductCount
with Result of concat(count($MT_IN_APP_PURCHASE/Root/Products/Product), ' products available')

After the products are queried, the app store sends back data about the queried products to the client device and this data is stored in the $MT_IN-APP_PURCHASE page source (*see Query Available Products*[1509] *for details*). The actions to send the request and store the data are carried out automatically by MobileTogether.

As part of the design of our example project, we have implemented two additional steps:

- The button *Get Available Products* specifies an Update Node[886] action, which counts the number of products now in the page source tree and updates the **$PERSISTENT/Root/ProductCount** node with this number. Since this node is the page source link of the label corresponding to the *Get Available Products* button, the number of products shown in this label is updated immediately.
- The products that are now stored in the page source tree are displayed in a dynamic table[1069], with each product being displayed in a row group of the table (*see screenshot below*). We have created two similar tables for displaying the products. The first table is for iOS and Windows products; the second table is for Android products. We have created two separate tables because the structure of the $MT_IN-APP_PURCHASE page source is different for Android; consequently, we need different XPath expressions to display the data of Android products. To display the appropriate table on each device, the *Visibility* property of the respective table tests whether the global variable[1300] **$MT_ANDROID** is **true()** or **false()**.



Available products

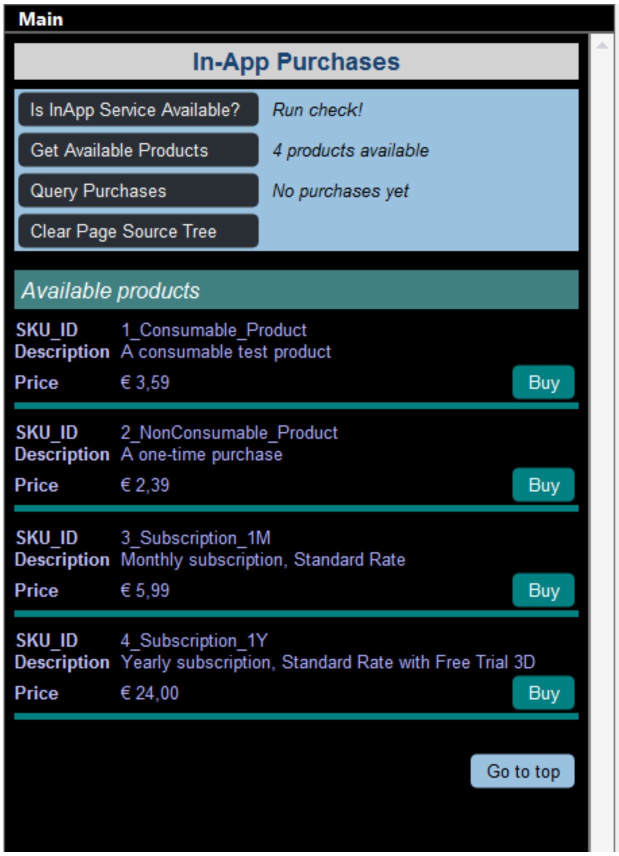
Product ($MT_IN_APP_PURCHASE)

| | | |
|---|---|---|
| SKU_ID | SKU_ID ($MT_IN_APP_PURCHASE) | |
| Description | Description ($MT_IN_APP_PURCHASE) | |
| Price | Price ($MT_IN_APP_PUR | XPath: let $sk |

Product ($MT_IN_APP_PURCHASE)

| | | |
|---|---|---|
| SKU_ID | SKU_ID ($MT_IN_APP_PURCHASE) | |
| Description | XPath: concat(@Description, ', ', Subscription/@BaseI | |
| Price | XPath: if (not(Subscription | XPath: let $sk |

The effect of both the actions listed above can be seen in the simulator screenshot shown below. When the button *Get Available Products* button is clicked, the four products that are stored in the XML data file[1517] are queried (in the case of an Android device simulation, these are the **`Product`** elements of the **`Android`** element). This product data is sent to the device and stored in the **`Product`** elements of the **`$MT_IN_APP_PURCHASE`** page source.



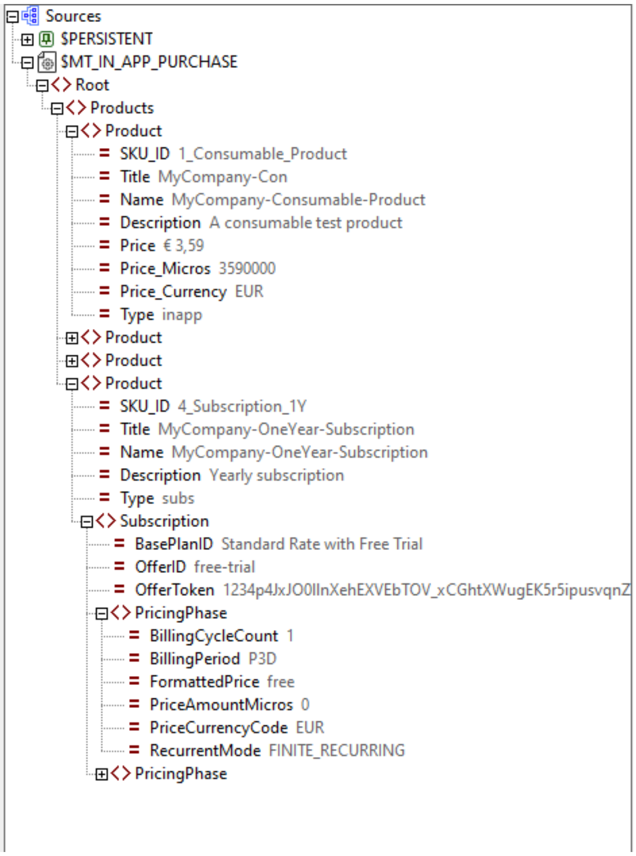
Main

In-App Purchases

| | |
|---|---|
| Is InApp Service Available? | *Run check!* |
| Get Available Products | *4 products available* |
| Query Purchases | *No purchases yet* |
| Clear Page Source Tree | |

*Available products*

| | | |
|---|---|---|
| **SKU_ID** | 1_Consumable_Product | |
| **Description** | A consumable test product | |
| **Price** | € 3,59 | Buy |
| **SKU_ID** | 2_NonConsumable_Product | |
| **Description** | A one-time purchase | |
| **Price** | € 2,39 | Buy |
| **SKU_ID** | 3_Subscription_1M | |
| **Description** | Monthly subscription, Standard Rate | |
| **Price** | € 5,99 | Buy |
| **SKU_ID** | 4_Subscription_1Y | |
| **Description** | Yearly subscription, Standard Rate with Free Trial 3D | |
| **Price** | € 24,00 | Buy |



```
Sources
  $PERSISTENT
  $MT_IN_APP_PURCHASE
    Root
      Products
        Product
          SKU_ID 1_Consumable_Product
          Title MyCompany-Con
          Name MyCompany-Consumable-Product
          Description A consumable test product
          Price € 3,59
          Price_Micros 3590000
          Price_Currency EUR
          Type inapp
        Product
        Product
        Product
          SKU_ID 4_Subscription_1Y
          Title MyCompany-OneYear-Subscription
          Name MyCompany-OneYear-Subscription
          Description Yearly subscription
          Type subs
          Subscription
            BasePlanID Standard Rate with Free Trial
            OfferID free-trial
            OfferToken 1234p4JxJO0lInXehEXVEbTOV_xCGhtXWugEK5r5ipusvqnZ
            PricingPhase
              BillingCycleCount 1
              BillingPeriod P3D
              FormattedPrice free
              PriceAmountMicros 0
              PriceCurrencyCode EUR
              RecurrentMode FINITE_RECURRING
            PricingPhase
```

After the two additional steps mentioned above have been carried out and the **`$MT_IN_APP_PURCHASE`** page source has been updated with the new **`Product`** data, the following happens:

- The count of products is updated, and
- Information about each product is displayed in a row of a dynamic table.

**Note:** The green label containing the *Available products* text has its `Visible` property set to the XPath expression **`$MT_IN_APP_PURCHASE/Root/Products/Product`**. This causes the label to be visible only when there is at least one **`Products/Product`** element.

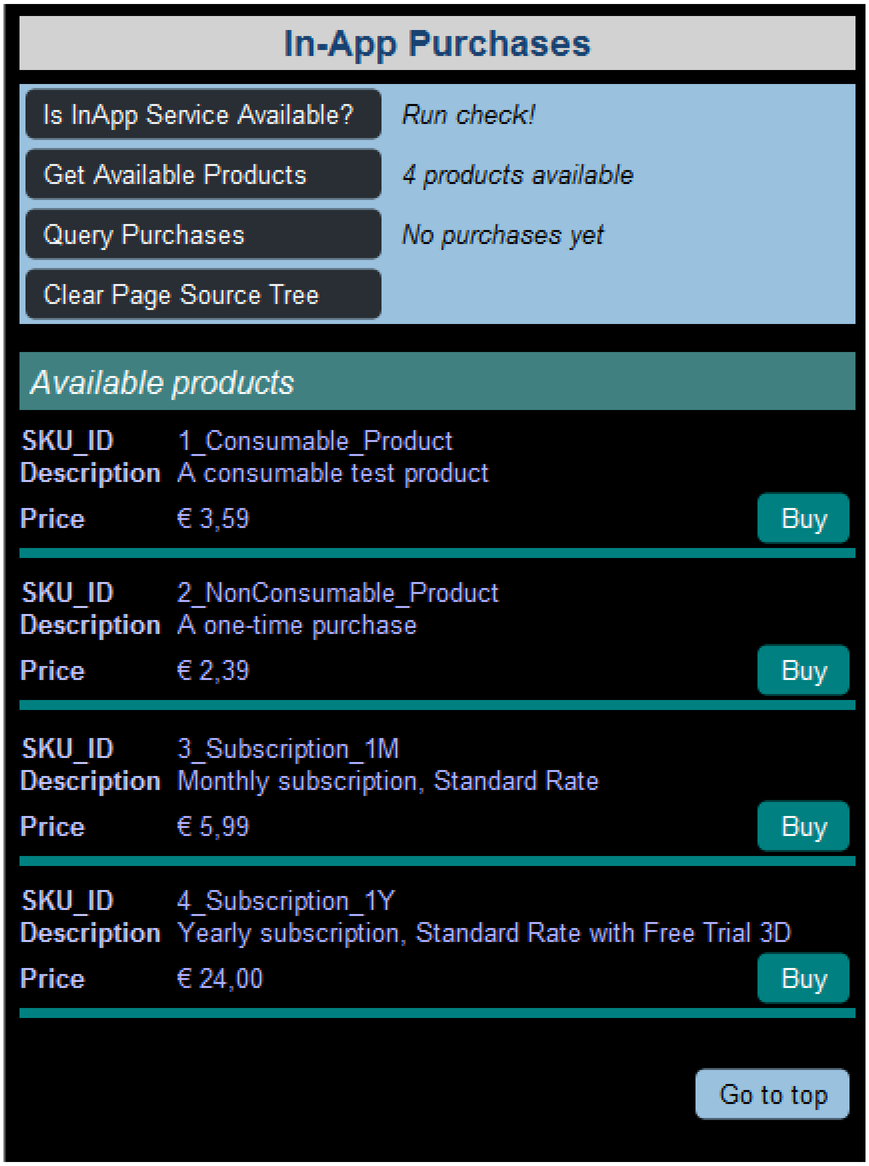
**Note:** The table itself will be visible only when it has content; so there is no need to set a condition for visibility.

In the next section 1524, we describe the **Buy** button that is displayed with each product.

## 24.7.5 The "Buy" Button

In our example project, the end user can get a list of products for in-app purchase 1521 by tapping the **Get Available Products** button (*see screenshot below left*). For each product in this list, there is a **Buy** button that the end user can use to buy the product (*screenshot below left*). The screenshot below right shows the design, in which the button can be seen at bottom right. The button has two important attributes: (i) its text, (ii) the purchase request that is sent to the app store.

These are described below specifically for **Android devices**. You might need to modify the design to suit a different app store's requirements.



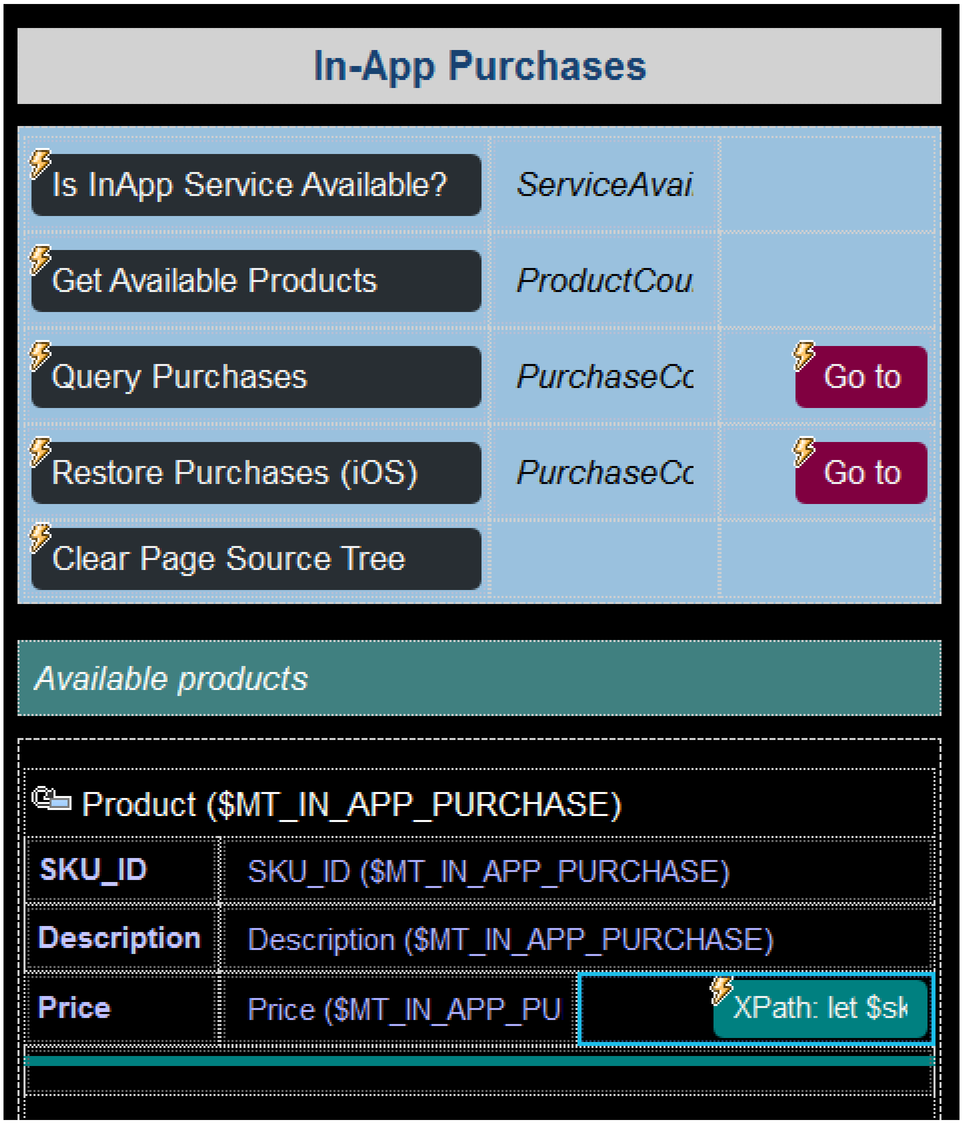
*Run check!*

*4 products available*

*No purchases yet*

*Available products*

| | |
|---|---|
| **SKU_ID** | 1_Consumable_Product |
| **Description** | A consumable test product |
| **Price** | € 3,59 |
| **SKU_ID** | 2_NonConsumable_Product |
| **Description** | A one-time purchase |
| **Price** | € 2,39 |
| **SKU_ID** | 3_Subscription_1M |
| **Description** | Monthly subscription, Standard Rate |
| **Price** | € 5,99 |
| **SKU_ID** | 4_Subscription_1Y |
| **Description** | Yearly subscription, Standard Rate with Free Trial 3D |
| **Price** | € 24,00 |



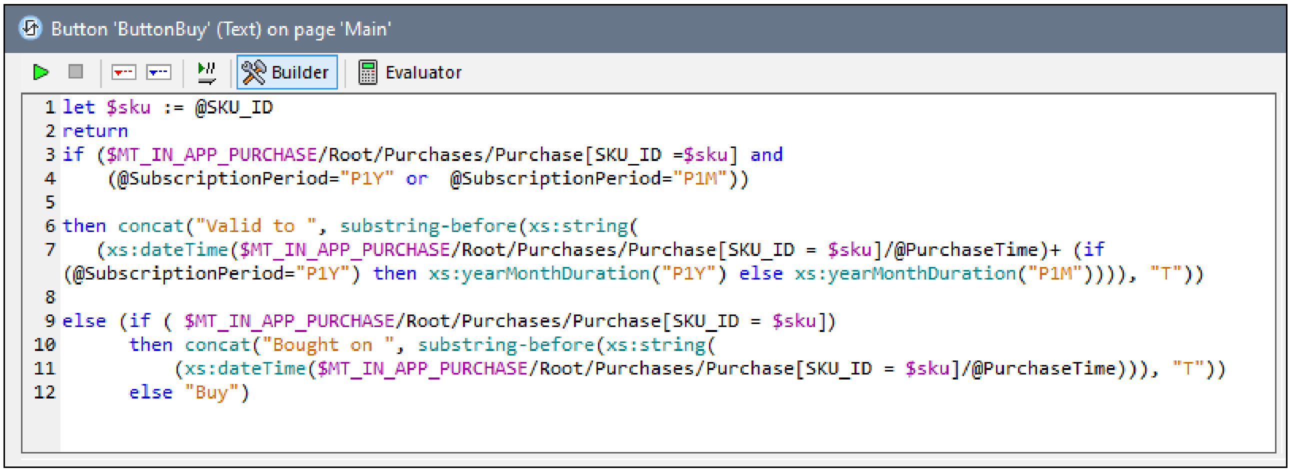
*Button text*
Since the **Buy** button is placed in a row group that repeats for each **`Product`** element in the `$MT_IN-APP_PURCHASE` page source, each displayed button applies to a specific **`Product`** element. What we want to do for the text of each button is this:

- If the product has not yet been bought, we want to display the text *Buy*. We can determine whether a product has been bought by checking if it exists as a **`Purchase`** element in the `$MT_IN-APP_PURCHASE` page source. This can be ascertained by checking the **`SKU_ID`** child element of **`Purchase`** elements to see if any matches the **`@SKU_ID`** attribute of the current **`Product`** element.
- If the product has been bought, we want to distinguish between (i) subscription products on one hand and (ii) non-subscription products on the other. For subscription products that have been bought, we want to display the date till which the subscription is valid. For non-subscription products that have been bought, we want to display the purchase date.
- *For Android:* Subscription products will have an attribute **`Subscription/PricingPhase/@BillingPeriod`**, so we can look for its existence to find out whether

the product is a subscription product. Further, if the subscription is yearly, then the value of the attribute will be `P1Y` (that is, **`@BillingPeriod="P1Y")`**; if it is monthly, then `P1M` (that is, **`@BillingPeriod="P1M"`**). Based on the value of the **`@BillingPeriod`** attribute, we can calculate the expiry date by adding the appropriate period (year or month) to the purchase date.

- *For iOS and Windows:* Subscription products will have an attribute **`@SubscriptionPeriod`**, so we can look for its existence to find out whether the product is a subscription product. Further, if the subscription is yearly, then the value of the attribute will be `P1Y` (that is, **`@SubscriptionPeriod="P1Y")`**; if it is monthly, then `P1M` (that is, **`@SubscriptionPeriod="P1M"`**). Based on the value of the **`@SubscriptionPeriod`** attribute, we can calculate the expiry date by adding the appropriate period (year or month) to the purchase date.
- If a non-subscription product (consumables and non-consumables) has been bought, we want to display the purchase date.

The XPath expression for the `Text` property of the **Buy** button (of the iOS and Windows table) will therefore be that shown in the screenshot below. Note that the **context node** of the XPath expression will be the **`$MT_IN_APP_PURCHASE/Root/Products/Product`** element. This is because the button is placed inside a dynamic table 1069 where rows repeat on the **`Product`** element. Note that the locator expressions in the XPath expression shown in the screenshot will be different for the Android table. This is because the relevant information, such as subscription period, are stored in different nodes of the data structure than those of the iOS/Windows data structure. (To see the corresponding XPath expression for Android, open it in the second table.)



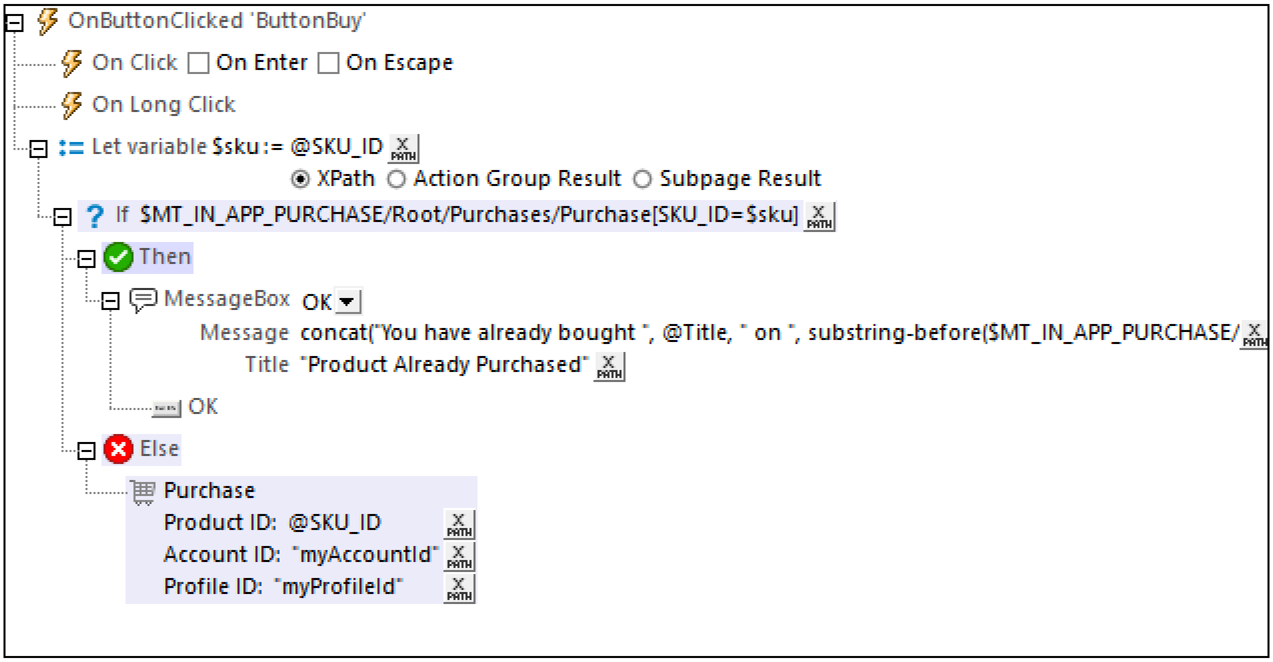
```
let $sku := @SKU_ID
return
if ($MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID =$sku] and
    (@SubscriptionPeriod="P1Y" or  @SubscriptionPeriod="P1M"))

then concat("Valid to ", substring-before(xs:string(
   (xs:dateTime($MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID = $sku]/@PurchaseTime)+ (if
(@SubscriptionPeriod="P1Y") then xs:yearMonthDuration("P1Y") else xs:yearMonthDuration("P1M")))), "T"))

else (if ( $MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID = $sku])
      then concat("Bought on ", substring-before(xs:string(
          (xs:dateTime($MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID = $sku]/@PurchaseTime))), "T"))
      else "Buy")
```

*Purchase request to app store*

Depending on the text that is displayed on the **Buy** button, there are two possible ways for the solution to react to a button-click event:

- If the product has already been bought, then do not proceed with the in-app purchase
- If the product has not yet been bought, then go ahead with the purchase request to the app store.

These two actions have been implemented via conditional processing for the button's on-click event, shown in the screenshot below.



```
OnButtonClicked 'ButtonBuy'
    On Click □ On Enter □ On Escape
    On Long Click
    Let variable $sku:= @SKU_ID
        ● XPath ○ Action Group Result ○ Subpage Result
    If $MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID=$sku]
        Then
            MessageBox OK
                Message concat("You have already bought ", @Title, " on ", substring-before($MT_IN_APP_PURCHASE/
                Title "Product Already Purchased"
                OK
        Else
            Purchase
                Product ID: @SKU_ID
                Account ID: "myAccountId"
                Profile ID: "myProfileId"
```

If the product has already been bought, a message to this effect is displayed. Otherwise, the Purchase[933] action is executed, and the app sends a purchase request for the current product to the app store.

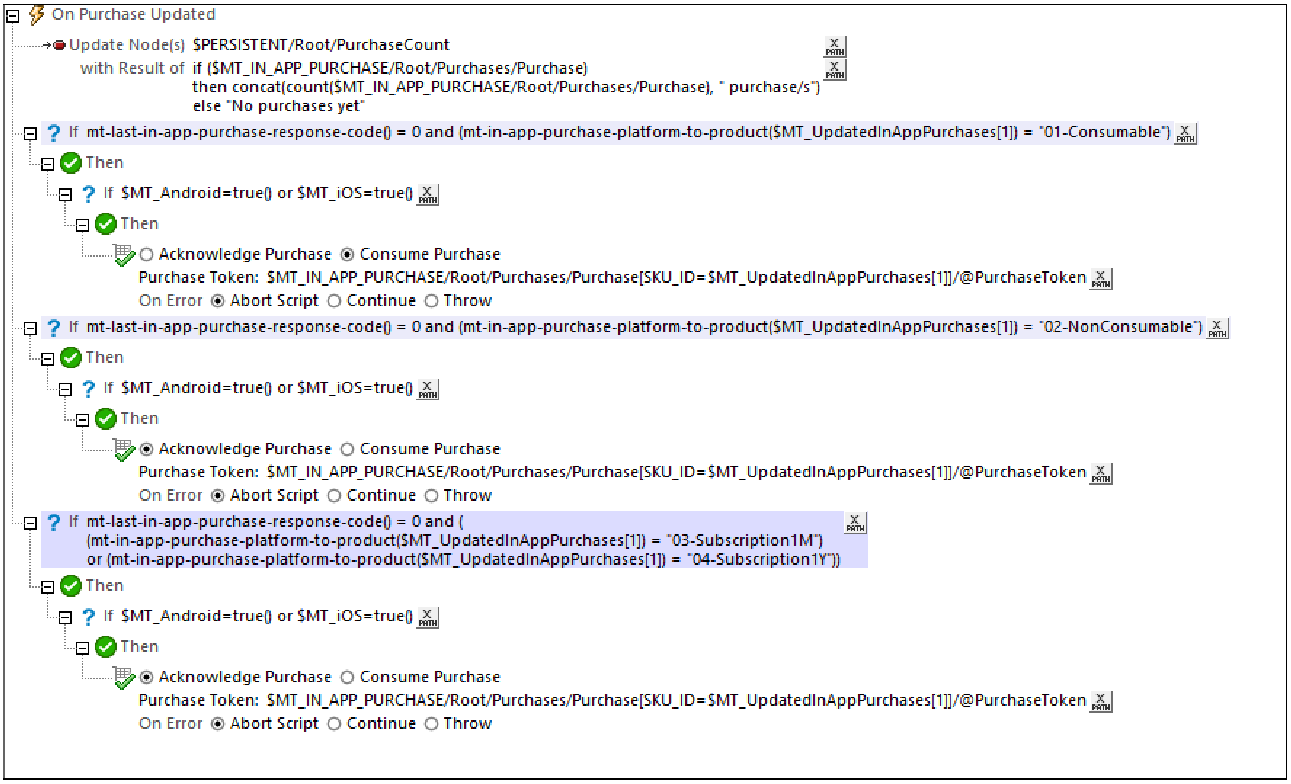
**Note:** The above scenario is a simplified one in that a product can only be bought once. Alternatively, you might want (i) to allow a product to be bought multiple times, or (ii) to check whether a subscription has expired or not before enabling/disabling a renewal.

## 24.7.6 The OnPurchaseUpdated Event

After an app store accepts a purchase request that has been made via the Purchase[933] action, it sends information about the accepted purchase to the app. This information is stored in a **`Purchase`** element of the `$MT_IN-APP_PURCHASE` page source so that the app can reference it. After this information has been received from the app store, you would most likely need to carry out certain actions. It is to enable the possibility of triggering actions automatically upon receiving such "purchase-accepted" information from the app store, that the `OnPurchaseUpdated` event has been created. To set up actions that are triggered by this event, go to the project property In-App Purchase Actions[296], which opens the Actions dialog for the `OnPurchaseUpdated` event.

In our example, we have defined two actions to be carried out automatically upon receiving a "purchase-accepted" response from the app store:

- Re-count the number of purchases and update the count that is displayed in the app.
- Send an app-side acknowledgment of the in-app purchase to the app store. An acknowledgment is commonly required by app stores to complete the transaction and forward to the product vendor the money that was paid for the product. In the design, you can acknowledge an in-app purchase by specifying an Acknowledge Purchase[936] action.



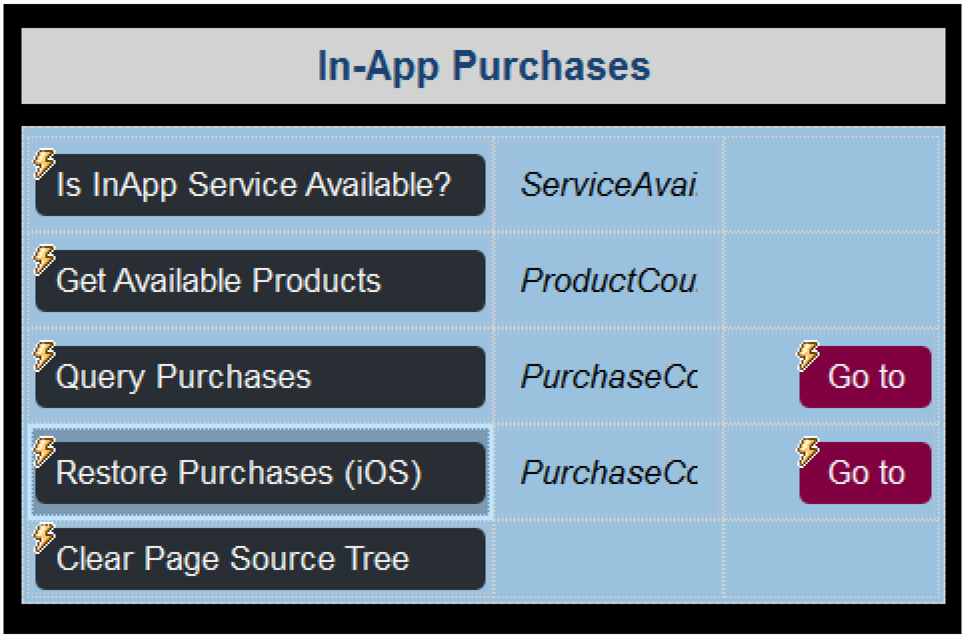
```
On Purchase Updated
  Update Node(s) $PERSISTENT/Root/PurchaseCount
    with Result of if ($MT_IN_APP_PURCHASE/Root/Purchases/Purchase)
                   then concat(count($MT_IN_APP_PURCHASE/Root/Purchases/Purchase), " purchase/s")
                   else "No purchases yet"
  ? If mt-last-in-app-purchase-response-code() = 0 and (mt-in-app-purchase-platform-to-product($MT_UpdatedInAppPurchases[1]) = "01-Consumable")
    Then
      ? If $MT_Android=true() or $MT_iOS=true()
        Then
          ○ Acknowledge Purchase ● Consume Purchase
          Purchase Token: $MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID=$MT_UpdatedInAppPurchases[1]]/@PurchaseToken
          On Error ● Abort Script ○ Continue ○ Throw
  ? If mt-last-in-app-purchase-response-code() = 0 and (mt-in-app-purchase-platform-to-product($MT_UpdatedInAppPurchases[1]) = "02-NonConsumable")
    Then
      ? If $MT_Android=true() or $MT_iOS=true()
        Then
          ● Acknowledge Purchase ○ Consume Purchase
          Purchase Token: $MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID=$MT_UpdatedInAppPurchases[1]]/@PurchaseToken
          On Error ● Abort Script ○ Continue ○ Throw
  ? If mt-last-in-app-purchase-response-code() = 0 and (
       (mt-in-app-purchase-platform-to-product($MT_UpdatedInAppPurchases[1]) = "03-Subscription1M")
       or (mt-in-app-purchase-platform-to-product($MT_UpdatedInAppPurchases[1]) = "04-Subscription1Y"))
    Then
      ? If $MT_Android=true() or $MT_iOS=true()
        Then
          ● Acknowledge Purchase ○ Consume Purchase
          Purchase Token: $MT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID=$MT_UpdatedInAppPurchases[1]]/@PurchaseToken
          On Error ● Abort Script ○ Continue ○ Throw
```

The screenshot above shows the `OnPurchaseUpdated` actions that have been defined in our example project. The acknowledgments have been made conditional on the specific product because (i) consumables and (ii) subscriptions/non-consumables require different types of acknowledgment.

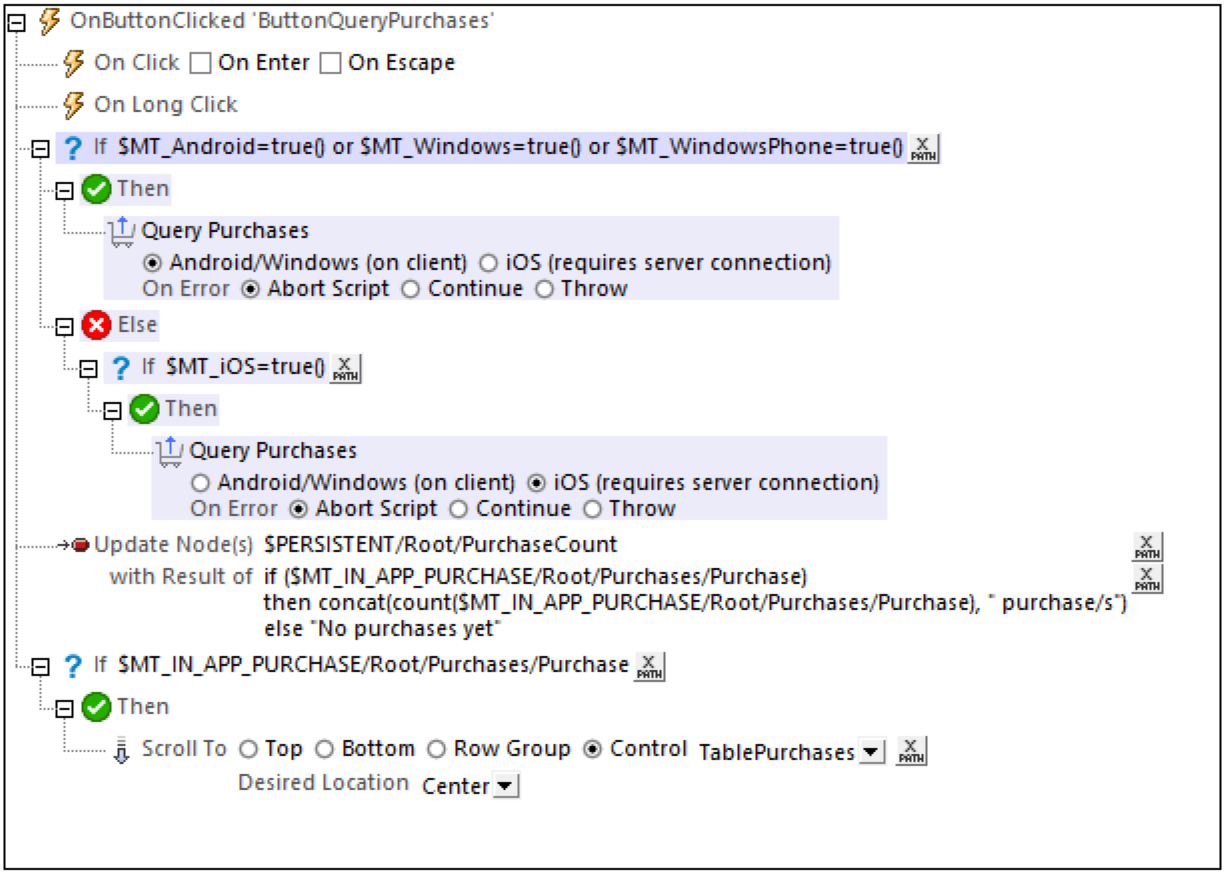
**Note:** You might need to modify the actions above to suit different app store requirements.

## 24.7.7 Query/Restore Purchases

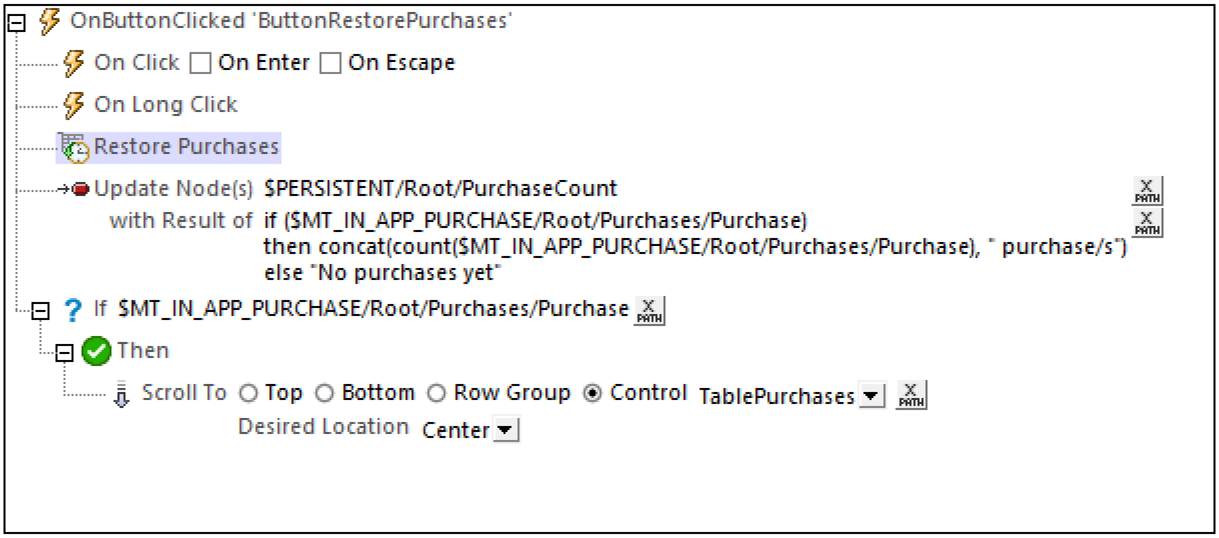
The Query Purchases and Restore Purchases (iOS) buttons retrieve information about the purchases of the current user from the app store. The buttons work by executing, respectively, the Query Purchases 934 action and Restore Purchases 934 action. The difference between the two actions is that the Restore Purchases 934 action, in addition to retrieving in-app purchases carried out from the app on the current device, also restores purchases made by the current user on other devices. This is a requirement of the iOS app store and has therefore entailed that a separate action be created for the purpose.



In our simulation, the two buttons both update the **`Purchases`** element of the `$MT_IN-APP_PURCHASE` page source with information relating to purchases that is fetched from the simulation data file 1517.



```
OnButtonClicked 'ButtonQueryPurchases'
  On Click ☐ On Enter ☐ On Escape
  On Long Click
  ? If $MT_Android=true() or $MT_Windows=true() or $MT_WindowsPhone=true()
    Then
      Query Purchases
        ● Android/Windows (on client) ○ iOS (requires server connection)
        On Error ● Abort Script ○ Continue ○ Throw
    Else
      ? If $MT_iOS=true()
        Then
          Query Purchases
            ○ Android/Windows (on client) ● iOS (requires server connection)
            On Error ● Abort Script ○ Continue ○ Throw
  Update Node(s) $PERSISTENT/Root/PurchaseCount
    with Result of if ($MT_IN_APP_PURCHASE/Root/Purchases/Purchase)
                   then concat(count($MT_IN_APP_PURCHASE/Root/Purchases/Purchase), " purchase/s")
                   else "No purchases yet"
  ? If $MT_IN_APP_PURCHASE/Root/Purchases/Purchase
    Then
      Scroll To ○ Top ○ Bottom ○ Row Group ● Control TablePurchases
                Desired Location Center
```



```
OnButtonClicked 'ButtonRestorePurchases'
  On Click ☐ On Enter ☐ On Escape
  On Long Click
  Restore Purchases
  Update Node(s) $PERSISTENT/Root/PurchaseCount
    with Result of if ($MT_IN_APP_PURCHASE/Root/Purchases/Purchase)
                   then concat(count($MT_IN_APP_PURCHASE/Root/Purchases/Purchase), " purchase/s")
                   else "No purchases yet"
  ? If $MT_IN_APP_PURCHASE/Root/Purchases/Purchase
    Then
      Scroll To ○ Top ○ Bottom ○ Row Group ● Control TablePurchases
                Desired Location Center
```

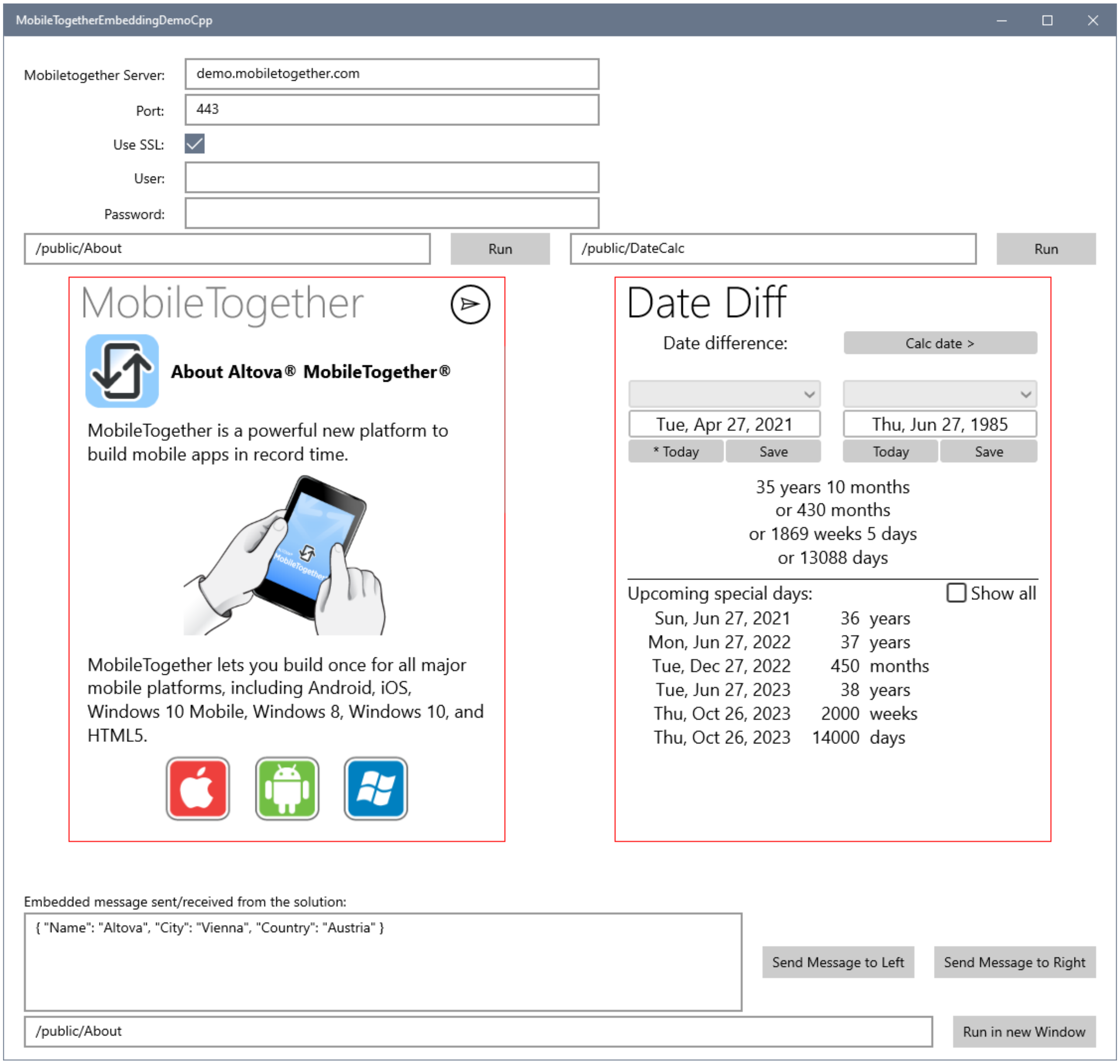
Additionally, note the following:

- The count of purchases is updated after the Query Purchases 934 action or Restore Purchases 934 action is carried out, and the count is automatically updated in the display.

- Since the Restore Purchases button should be available only on iOS devices, the row containing the button has its visibility made conditional on the device being an iOS device. This has been done by setting the rows's `Visible` property to check the value of the static variable `$MT_iOS`.

# 25 MT Solutions in UWP Apps

You can integrate MobileTogether solutions in Universal Windows Platform (UWP) apps. Each MobileTogether solution is placed in a UWP app via a SolutionView control, which Altova has introduced specially for this purpose. For example, the screenshot below shows the window of a UWP app which contains two SolutionView controls (outlined in red), each of which displays a MobileTogether solution. Each UWP app can integrate more than one solution (as in the screenshot below), all of which can run at the same time. Additionally, a UWP app can start a solution in a new UWP window.



MobileTogetherEmbeddingDemoCpp

Mobiletogether Server: demo.mobiletogether.com

Port: 443

Use SSL: ☑

User:

Password:

/public/About

/public/DateCalc

MobileTogether

About Altova® MobileTogether®

MobileTogether is a powerful new platform to build mobile apps in record time.

MobileTogether lets you build once for all major mobile platforms, including Android, iOS, Windows 10 Mobile, Windows 8, Windows 10, and HTML5.

Date Diff

Date difference:

Tue, Apr 27, 2021

Thu, Jun 27, 1985

35 years 10 months
or 430 months
or 1869 weeks 5 days
or 13088 days

Upcoming special days: □ Show all

| | | |
|---|---|---|
| Sun, Jun 27, 2021 | 36 | years |
| Mon, Jun 27, 2022 | 37 | years |
| Tue, Dec 27, 2022 | 450 | months |
| Tue, Jun 27, 2023 | 38 | years |
| Thu, Oct 26, 2023 | 2000 | weeks |
| Thu, Oct 26, 2023 | 14000 | days |

Embedded message sent/received from the solution:

{ "Name": "Altova", "City": "Vienna", "Country": "Austria" }

/public/About

The broad procedure for integrating MT solutions in your UWP app is as follows:

1. Reference the MT Libraries 1536: Copy the relevant MobileTogether libraries to your project’s directory, and reference 1536 the MobileTogether libraries in your project.
2. Add one or more SolutionView controls 1537 to your XAML page 1537 or control.
3. Create the code 1539 to manipulate the SolutionView control programatically and to listen for events.
4. To interact with a MT solution hosted by SolutionView control, you can use MobileTogether's `EmbeddedMessage` mechanism in your code. See Messaging in the UWP App 1540 for information about how to send and receive messages from the MT solution.
5. Compile the UWP app.

This section is organized along the lines of the procedure given above.

For an example, see the topic Example UWP App 1542.

## 25.1 Reference the MT Libraries

In order to integrate MobileTogether solutions in your UWP app, you will need to reference the relevant MobileTogether libraries from your UWP project directory. These libraries will be located in a descendant folder of your MobileTogether application folder, so MobileTogether Designer will need to have been installed. Create the library reference as follows:

1. Inside the application folder[60], go to the subfolder `MobileTogetherSPL\WindowsApp`. The required libraries are in the ZIP file `CustomAppTemplateWindows10.zip`.
2. Copy the contents of the ZIP folder to your UWP project directory. The required files and folders are:

   - `MobileTogether` *(folder)*
   - `zxing.uwp.ARM` *(folder)*
   - `zxing.uwp.x64` *(folder)*
   - `zxing.uwp.x86` *(folder)*
   - `MobileTogether.winmd`
   - `MobileTogether.pri`
   - `MobileTogether.Windows.ARM.dll`
   - `MobileTogether.Windows.x86.dll`
   - `MobileTogether.Windows.x64.dll`

3. Add a reference to the MT libraries listed above, the best way being to manually edit the `.vcproj` file as listed below. Note that each implemented platform requires a different DLL.

```
<ItemGroup>
    <Reference Include="MobileTogether">
       <HintPath>MobileTogether.winmd</HintPath>
       <Implementation Condition=" '$(Platform)' == 'ARM'
">MobileTogether.Windows.ARM.dll</Implementation>
       <Implementation Condition=" '$(Platform)' == 'Win32'
">MobileTogether.Windows.x86.dll</Implementation>
       <Implementation Condition=" '$(Platform)' == 'x64'
">MobileTogether.Windows.x64.dll</Implementation>
    </Reference>
</ItemGroup>
```

For an example, see the topic Example UWP App[1542].

# 25.2 XAML File

To add a SolutionView control to your XAML file, do the following:

1. Declare the MobileTogether namespace in your XAML page or control like this:

```
<Page...
      xmlns:mobiletogether="using:MobileTogether">
```

2. Add the SolutionView control:

```
<mobiletogether:SolutionView
   IsAutoSuspendResumeEnabled="False"
   IsBackKeyEnabled="False"
   IsEscapeKeyEnabled="False"
   IsEnterKeyEnabled="False"
   ServerURL="demo.mobiletogether.com"
   Port="443"
   UseSSL="True"
   User=""
   Password=""
   SolutionURL="/public/MyCollections?Par1=123&amp;Par2=456"
   EmbeddedMessage="MyEmbeddedMessage"
   SolutionFinished="OnSolutionFinished" />
```

For an example, see the topic Example UWP App 1542.

## Properties of the SolutionView control

The control's properties are listed below:

*IsBackKeyEnabled, IsEscapeKeyEnabled, IsEnterKeyEnabled*
Since multiple SolutionView controls can run in a single UWP app window, all running solutions would process the same **Back**, **Esc**, and **Enter** keys. To ensure that these keystrokes are not processed for the wrong solution, these three attributes enable these keys to be disabled for any given SolutionView control.

*ServerURL, Port, UseSSL*
The URL and port of the MobileTogether Server on which the solution has been deployed. The *UseSSL* attribute specifies whether the server connection uses SSL or not.

*User, Password*
Specifies the credentials of the user that is accessing the solution. To specify that access is anonymous, use the empty string as the value of both attributes.

*SolutionURL*
Specifies the path of the solution on MobileTogether Server. In the listing above, notice how **`SolutionURL`** can contain a solution's input parameters.

*EmbeddedMessage*
The embedded message that is sent from the SolutionView control to MobileTogether Server. The data must be sent as a serialized JSON object. (If your data is in XML format, then the XML document must be wrapped inside a JSON object.)

*SolutionFinished*
When the solution is finished or closed, the `SolutionFinished` event is fired. You can use this event, for example, to close the UWP app window after the solution finishes.

## 25.3 Run MT Solution from Code

To run a solution, you can specify the required properties of the SolutionView control in the XAML file[1537] itself. Alternatively, you can run a solution from code (typically C++ or C#) as follows.

You can instantiate a SolutionView control in the XAML file with a name, as follows:

```
<mobiletogether:SolutionView Name="x:MySolutionView" />
```

You can then run the solution from code as follows:

```
mySolutionView.ServerURL = "demo.mobiletogether.com";
mySolutionView.Port = "443";
mySolutionView.UseSSL = true;
mySolutionView.User = "";
mySolutionView.Password = "";
mySolutionView.SolutionURL = "/pulic/MyCollections?Par1=123&Par2=456";
await mySolutionView.StartSolution();
```

In code, the following actions can be performed via the corresponding methods listed below :

Page submit: `mySolutionView.Submit();`
Page refresh: `mySolutionView.Refresh();`
Go back: `mySolutionView.GoBack();`
Stop the solution: `mySolutionView.StopSolution();`
Suspend the solution: `mySolutionView.SuspendSolution();`
Access the solution's local storage: `mySolutionView.GetSolutionLocalFolder();`

Note the following points:

- The `SolutionURL` method can take the solution's input parameters.
- When the solution is finished or closed, the `SolutionFinished` event is fired. You can use this event, for example, to close the UWP app window after the solution finishes.
- Communication from the solution is carried out by writing to files in the solution's local storage or to any other accessible folder such as the Music, Videos, and Pictures folders.
- While it is possible to run the same solution in two separate SolutionView controls on the same page, there would be difficulties accessing the same resources. For example, you would encounter problems if you try to access a file in one SolutionView control while trying to delete it in another.

For an example, see the topic Example UWP App[1542].

# 25.4 Messaging in the UWP App

The broad stages for messaging would be as follows:

1. Data is communicated from the UWP app to the solution (displayed in a SolutionView control of the UWP page).
2. The solution sends the data to the solution's workflow on MobileTogether Server, where it is processed in the usual way.
3. Results are returned to the SolutionView control, where they can be (i) displayed as part of the solution, and/or (ii) passed back to the UWP app for display or further processing.

## Messaging mechanism

A message that is sent from a UWP app to the server is called an **embedded message**. The messaging mechanism uses three MobileTogether features:

- The **OnEmbeddedMessage** 397 event, which is available as a page event;
- The `$MT_EMBEDDEDMESSAGE` JSON page source, which is generated automatically if at least one action is defined for the **OnEmbeddedMessage** 397 event;
- The Embedded Message Back 924 action, which sends a serialized JSON string as a return message;

How these features are used for messaging in the UWP app is as follows:

- SolutionView control now has a method **ProcessEmbeddedMessage(msg)** that (i) populates the `$MT_EMBEDDEDMESSAGE` JSON page source, and (ii) triggers the page event **OnEmbeddedMessage** 397.
- SolutionView control now has an event named **EmbeddedMessage** that is triggered when, in the hosted solution, the Embedded Message Back 924 action is executed. This event can be processed synchronously or asynchronously, for example, by using the standard UWP deferral mechanism as listed below.

```
void EmbeddedMessage(object sender, EmbeddedMessageEventArgs e)
{
var deferal = e.GetDeferral();
... Do some parallel processing of e.Message ...
deferal.Complete();
}
```

## Messaging: step-by-step

The steps given below provide a detailed look at the messaging process.

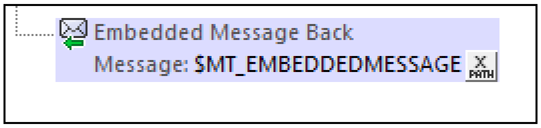
*UWP app to solution to MT Server*

1. The **ProcessEmbeddedMessage(msg)** method in your code 1539 can be used to transfer data from the UWP app to the SolutionView control. The data must be sent as a serialized JSON object. (If your data is in XML format, then the XML document must be wrapped inside a JSON object.)
2. The targeted solution must have an **OnEmbeddedMessage** 397 event that has at least one action defined for it. In this case, when an embedded message reaches the solution, a `$MT_EMBEDDEDMESSAGE` JSON page source is created automatically and populated. The page source will have a root element named `json`, and its structure and content will come from the embedded message.

3. On the server, the `$MT_EMBEDDEDMESSAGE` JSON page source can be processed via actions of the **`OnEmbeddedMessage`** 397 event or in any other way. This completes the first half of the transfer, from UWP app to MobileTogether Server. Processed data can now be sent back to the solution.

*MT Server to solution to UWP app*

4. The Embedded Message Back 924 action sends a return message to the SolutionView control in the form of a serialized JSON string. In defining the action, you can specify the message that you want to send. For example, in the screenshot below, the action specifies that the message to be sent back is the entire data structure stored in the `$MT_EMBEDDEDMESSAGE` JSON page source: that is, the tree's **`json`** node and its contents.



Embedded Message Back
Message: $MT_EMBEDDEDMESSAGE

5. When the solution's Embedded Message Back 924 action is executed, the SolutionView control's **`EmbeddedMessage`** event is triggered. The embedded message received by the SolutionView control can now be processed by the code of your UWP app.

## 25.5 Example UWP App

Your MobileTogether Designer installation contains an example C# project that shows how to use the SolutionView control in a simple UWP app. This project is available in the *(My) Documents* folder: `Altova\MobileTogetherDesigner10\MobileTogetherDesignerExamples\WindowsClientControl`.

To run this sample UWP app, do the following:

1. Go to the `C:\Program Files (x86)\Altova\MobileTogetherDesigner10\MobileTogetherSPL\WindowsApp` folder located in the MobileTogether Designer application folder and copy the MobileTogether libraries in this folder.
2. Go to the folder where the C# project is located and unzip the MobileTogether libraries there.
3. Start Visual Studio (2017 or newer) with administrator rights.
4. Load the C# project. Compile it and run it.

# 26 Server Services

A service is a set of MobileTogether actions that is deployed to **MobileTogether Server Advanced Edition** as a solution (`.mtd` file). The service is executed on the server when a specified set of MobileTogether Server conditions is met. These are the triggers that run the service, and they are defined in the administrator interface of MobileTogether Server Advanced Edition.

The broad procedure for creating and running a MobileTogether service is as follows:

1. Create a service[1544] in MobileTogether Designer. Each service is stored in a single `.mtd` file.
2. Deploy the service[1547] from MobileTogether Designer to MobileTogether Server Advanced Edition.
3. In the MobileTogether Server administrator interface, define the conditions that must be met in order to trigger the service[1548].

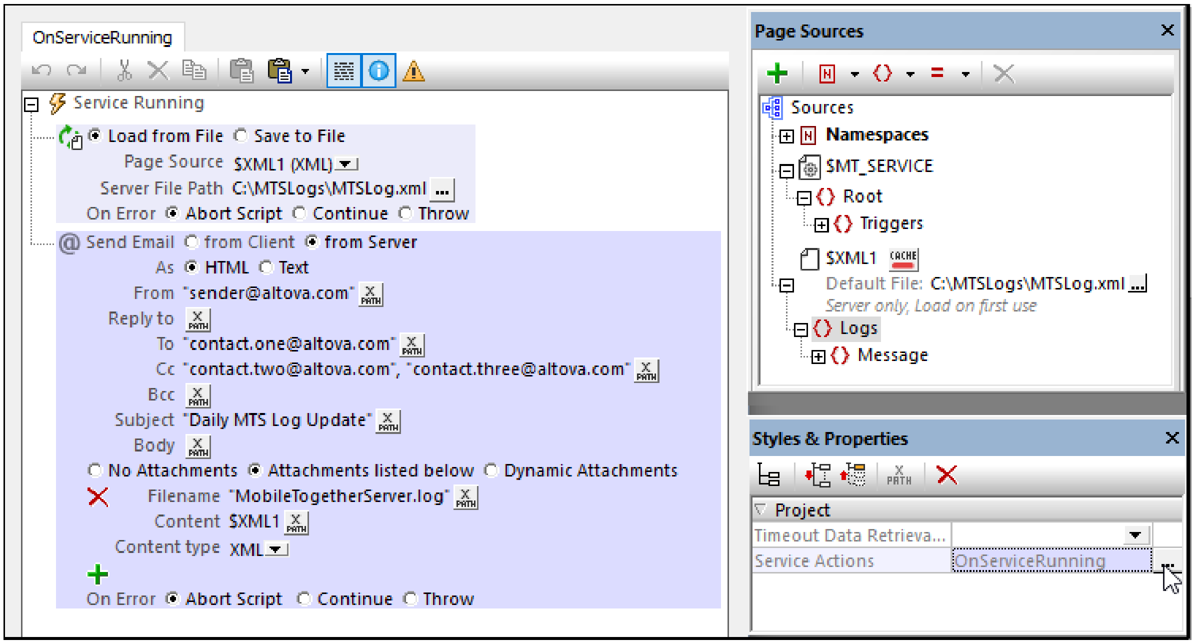
Each of these steps is described in more detail in the sub-sections of this section. For information about setting up a service on MobileTogether Server, see the MobileTogether Server documentation.

## 26.1 Creating a Service

A service is created in the same way as you create a solution. You can define page source trees 315 so that the service can use data from these trees. However, since a service is intended to run on the server (and so with no user interface), all controls 403 and the addition of new pages (whether top or sub 257) are disabled. The services you can run on the server are server-side actions, such as sending an email from the server or updating a node in a page source. So a set of such actions can be defined as the actions of a service; other actions are disabled.

To create a new service, do the following:

1. Click **File | New Service** 1563 to open a design file for the service. A new **service design** is created, and a `$MT_SERVICE` 1545 page source is automatically created. The MobileTogether Designer interface will look the same as that for a solution. One difference you might quickly notice is that no client-interface design is possible since all controls are disabled. Instead, all the actions you want to define for the service must be defined in the tab of the project event `OnServiceRunning`.
2. If you need to use page sources, add them to the Page Sources Pane 270. In the screenshot below, an XML page source named `$XML1` has been added.
3. Open the Actions dialog of the service (*see screenshot below*) in one of the following ways: (i) Click the **Service Action Tree** button located in the middle of the design page; or (ii) In the Styles & Properties Pane 274, click the **Additional Dialog** button of the **`Service Actions`** property.



OnServiceRunning

Service Running

● Load from File ○ Save to File
Page Source $XML1 (XML)
Server File Path C:\MTSLogs\MTSLog.xml
On Error ● Abort Script ○ Continue ○ Throw

Send Email ○ from Client ● from Server
As ● HTML ○ Text
From "sender@altova.com"
Reply to
To "contact.one@altova.com"
Cc "contact.two@altova.com", "contact.three@altova.com"
Bcc
Subject "Daily MTS Log Update"
Body
○ No Attachments ● Attachments listed below ○ Dynamic Attachments
Filename "MobileTogetherServer.log"
Content $XML1
Content type XML
On Error ● Abort Script ○ Continue ○ Throw

Default File: C:\MTSLogs\MTSLog.xml
Server only, Load on first use

Timeout Data Retrieva...
Service Actions OnServiceRunning

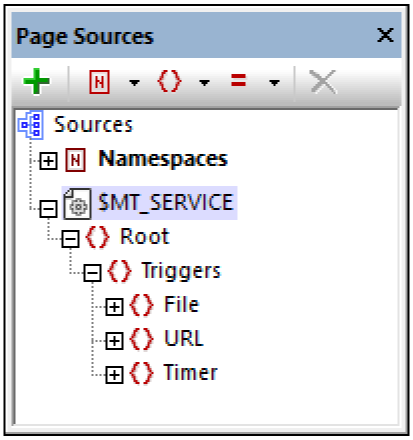
4. The left-hand pane of the dialog (*not shown in screenshot above*) displays all the available actions for services. The unavailable actions are disabled (and shown grayed out). Drag the actions you want to execute as the service into the `OnServiceRunning` tab. These actions constitute the **Service Action Tree**. In the screenshot above, two actions have been added: (i) *Load from File* loads the MobileTogether Server log file to the `$XML1` page source, and (ii) *Send Email* sends emails to three recipients, with the `$XML1` tree of the MobileTogether Server logs as an attachment.
5. Click **OK** to finish creating the actions of the service.
6. Save the file (**Ctrl+S**) with a suitable name for the service and a filetype of `.mtd`.

**Note:** You can create only one set of actions for each service. If you select **File | New Service** 1563 a second time, a new empty service file is created.

**Note:** You can localize services for other languages via the Localization dialog 1600 (**Project | Localization** 1600 menu command).

## The $MT_SERVICE page source

The `$MT_SERVICE` page source is automatically created when the service design is created. The screenshot and listing below show the structure of the page source.



⊟ *Structure of $MT_SERVICES page source*

```
<Root>
   <Triggers>
      <File name="" filename="" reason=""/>
      <URL name="" url=""/>
      <Timer name=""/>
   </Triggers>
</Root>
```

At run time, data about the triggers that have been set for the service will be passed from the server to the page source and will be stored in appropriate nodes of the page source. For example, the name of the file that activates a File System trigger will be stored in the **`//File/@filename`** node of the page source. If the XPath expressions of service actions access these nodes, then the run time information stored in these nodes can be used by the XPath expressions. For example, the name of the file that triggered a service action can be sent in a Send Email To 693 action, together with the reason for the trigger being activated (new file created, file modified, or file deleted)—all of which is information that cannot be known beforehand, but only at run time.

Since the relevant nodes of the page source will be automatically filled at run time, there is nothing further that you need to do concerning the **`$MT_SERVICE`** page source or the populating of its nodes. Its use to you is as a source of (additional) run time information about server-side triggers. You can access this information via XPath expressions, and use it: (i) to make service actions conditional on the value of the information, and/or (ii) as data to be passed on in a service action.

**Note:** For simulations, you can enter data in a **`$MT_SERVICE`** page source that will be used exclusively for simulations. This data simulates the data received at run time. How to create a **`$MT_SERVICE`** page source for simulations is described in the topic Service Trigger Simulations 1383.

## Service properties

In the Styles & Properties Pane[274] (*see screenshot above*), you can set a data retrieval timeout for the service (in seconds).

This is the amount of time the server waits for data to be retrieved from a source external to the server (from a DB or URL, for example). The value is an integer value in seconds that can be entered or selected from the dropdown list of the combo box. The default value is 10 seconds. If the timeout period is exceeded, then an error message is displayed. An exception to this is when load actions have the setting *On error* set to `Continue`. In this case, the *On Error* actions of that action's `Continue` setting are executed.

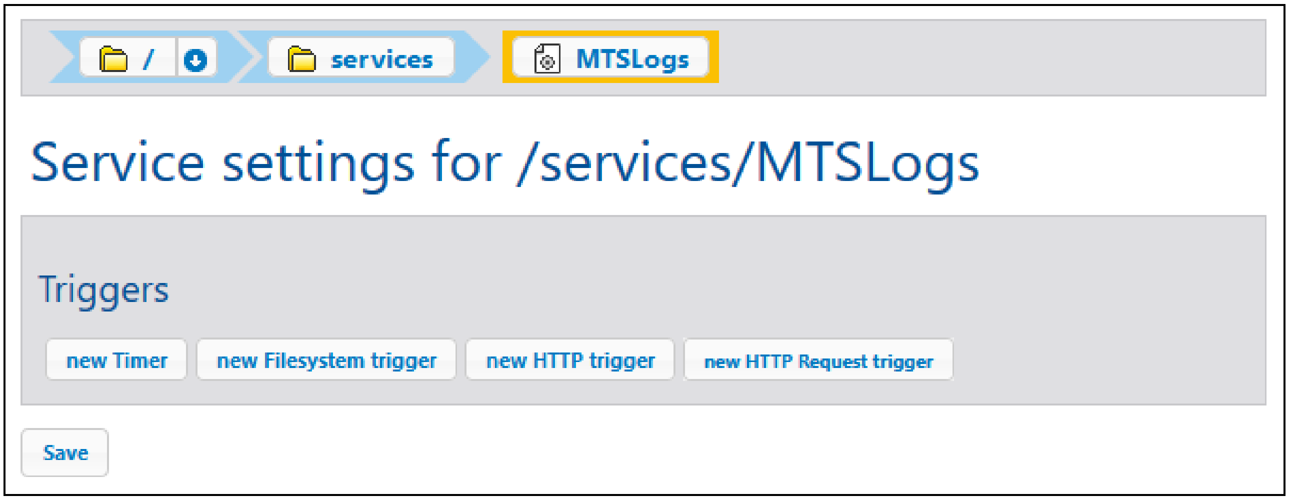
## 26.2 Deploying a Service

After a service file has been saved, you must deploy it to MobileTogether Server Advanced Edition. Do this in the same way you would deploy any solution 92, by using the **File | Deploy to MobileTogether Server** 1574.

**Note:** You can deploy MTS service files to **MobileTogether Server Advanced Edition** only. Trying to deploy to a standard edition will result in an error.

# 26.3 Running a Service

After a service has been deployed to MobileTogether Server Advanced Edition, it is listed in the Workflows tab, from where you can access the service's configuration (or settings) interface. The service's configuration (or settings) interface enables you to define and manage the triggers that run the service (*see screenshot below*).



You can create the following types of triggers:

- *Timer triggers*, which enable you to specify at what time and with what frequency within a specified period you want the service to run.
- *File system triggers*, which enable you to trigger a service by checking for changes to a file or directory on the server.
- *HTTP triggers,* which enable you to trigger a service by checking for changes to a resource at a specified URI location.
- *HTTP Request triggers,* which enable you to trigger a service by making an HTTP request to the service. This way of starting a service is described in the topic Start Service via URL 1549.

For more information about how to access the service's settings interface and how to set triggers for the service, see the MobileTogether Server documentation.

# 26.4 Start Service via URL

You can start a service via its URL and pass parameter values to the service via the URL's query parameters. To start a service via its URL, you must, on MobileTogether Server, set an HTTP Request trigger[1548] for the service and make sure that the trigger is enabled. For information about how to do this, see the MobileTogether Server documentation.

We will explain how to start a service via an HTTP Request with an example. Let us say we want to pass the IP address of a mobile device to a MobileTogether Server service. (We might want to do this, for example, so that the service can store the IP address in a file on the MobileTogether Server and, in this way, make the IP address available to solutions that have access to the server.)

The steps of the mechanism are as follows:

1. A device connected to the Internet sends an HTTP **GET** request to the MobileTogether Server service. This will start the service.
2. The URL of the request contains, as one of its query parameters, the IP address that we want to store on the MobileTogether Server.
3. When the MobileTogether Server service receives the request it *automatically* stores all the query parameters of the URL in its **$MT_InputParameters**[1300] variable.
4. The IP address is obtained from the **$MT_InputParameters**[1300] variable and stored in a file on the MobileTogether Server, from where it will be available to a solution running on an MobileTogether Client device.

These steps are discussed in more detail below.

## URL of the HTTP GET request

The pattern of the URL must be as follows:

```
https://<mt-server-hosting-the-solution>:<client-port>/runservice?d=<path-to-
service>&<param1>=<value>
Example: https://localhost:8083/runservice?
d=/services/MyIPAddressService&ipaddress=someAddress
```

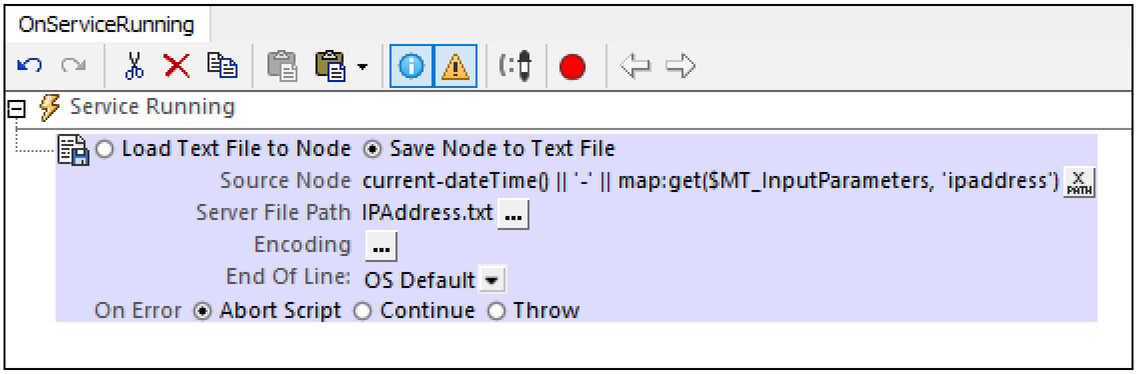
Note the following points:

- Ensure that the service solution is run as anonymous so that it can be called by any mobile device.
- Anonymous execution is only possible on the client port of the server (not the administrator port).
- Use `runservice?d=` to specify the path to the service solution on the server.
- Use `&key=value` to supply the name and value of a parameter, for example: `&ipaddress=someAddress`.
- You can supply multiple parameters. Add a new `&key=value` for each new parameter (with no space separator between parameters).

## Save IP address from $MT_InputParameters to a file on the MT server

When the service solution[1543] is started on the server by the HTTP **GET** request, the query parameters in the URL of the **GET** request are automatically stored in the **$MT_InputParameters**[1300] variable (*also see OnServerDeployment Input Parameters*[1574]).

The values stored in the **`$MT_InputParameters`** 1300 variable can now be accessed by the actions of the Service Action Tree 1544. For example, we can use a Save Text File 821 action as shown in the screenshot below. This action generates a text string that contains the IP address it obtains from the **`$MT_InputParameters`** 1300 variable, and then saves the generated string to a file on the server.



○ Load Text File to Node ● Save Node to Text File
Source Node current-dateTime() || '-' || map:get($MT_InputParameters, 'ipaddress')
Server File Path IPAddress.txt
Encoding
End Of Line: OS Default
On Error ● Abort Script ○ Continue ○ Throw

Note the following points about the screenshot example above:

- The string that contains the IP address is constructed as the value of *Source Node*. It is a concatenation of (i) the current dateTime and (ii) the IP address.
- The IP address is obtained from the value of the **`ipaddress`** key of the map stored in the **`$MT_InputParameters`** 1300 variable.
- The string generated for *Source Node* is saved to the **`IPAddress.txt`** file, which is located in the ServerSide Solution's Working Directory 1574.

The data in the text file can now easily be loaded into a page source node by using the Load Text File 821 action.

**Note:** You could also generate a string in XML format and save the string to an XML file. In this case, you would use the Load File 809 action to load data from the XML file.

# 27 Server Action Libraries

A server action library is a **.mtd** file that contains one or more Action Groups 940. Its functionality is limited to processing these Action Groups. At runtime, a solution can send a call to a server action library, with or without parameters. The server action library processes the specified Action Group and returns the result to the calling solution.

The advantage of using a server action library is that, since it operates independently of the main (calling) solution, it can be modified outside the solution. The benefits of this are:

- There is no need to modify the design of the main solution;
- Consequently, there would be no need to deploy a modified main solution on the server (to update the existing solution on the server);
- There is no need for client devices to download the modified solution;
- If you modify a server action library, all that needs to be done is to deploy the server action library to the server so that it overwrites the old server action library. This is a deployment that does not affect the overall runtime workflow.

Note, however, that the overheads (in processing power and time) will be greater when using a an Action Group in a server action library than when using an Action Group in the main solution. Consequently, a server action library should only be used if its main benefit (of independent modification and execution) provides a clear advantage over using an Action Group in the main solution.

## MobileTogether Server Advanced Edition

Server action libraries can be deployed to MobileTogether Server Advanced Edition only, not to the standard edition of MobileTogether Server.

## Use case

For example, consider a situation in which data is retrieved from a data structure that is outside your control and that changes arbitrarily and at irregular intervals. It might be difficult to modify and deploy a new main solution and update already downloaded solutions each time this data structure changes. However, a server action library could deal with the modifications of the data structure outside the main solution, process the modified data structure, and return data to the main solution so that the returned data conforms to the structure expected by the solution.

The advantages are clear:

- The server action library, since its purpose is limited, is smaller and simpler than the main solution and can be changed easily and quickly.
- The modified external data structure can be processed entirely within the server action library so that data is retrieved from the external data structure and sent to the main solution in a structure that the main solution uses.
- The modified server action library can be deployed to the server in a simple deployment procedure. The main solution, which is also deployed on the server, does not need to be modified in any way. Neither will solutions already downloaded to clients need to be updated.

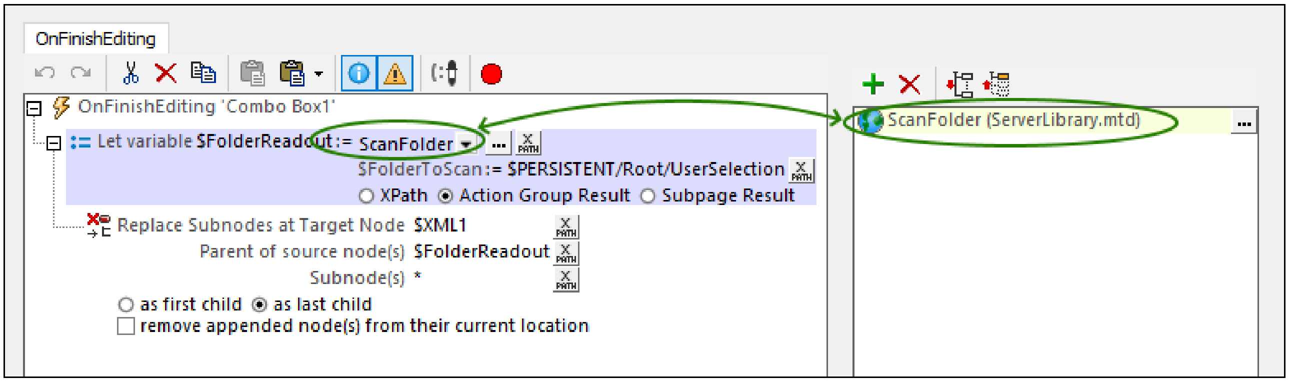
## Example

In our description of server action libraries in this section, we use the example of a simple server action library which reads the contents of a specified folder by using MobileTogether's Read Folder[847] action. This action in the server action library reads information about each item in the folder—subfolder or file—and stores information about the item in an XML tree of the server action library. The data tree is then sent from the server action library to the main solution as the return value of the server action library.

In our example, the key issue to be resolved in the main solution is how to specify which one of three alternative folders should be read. Let us say that the main solution works with three fixed (static and unchanging) datasets that are based on actual collections—one each for books, films, and music. In the main solution, the user selects one of the folders (books, films, or music), and the main solution must read the selected folder. A problem arises if any of the actual underlying collections is changed—because the main solution uses fixed datasets and these will then be outdated. A change could be any of the following: (i) new or modified folder contents, (ii) a modified structure of the data tree containing folder information, and/or (iii) a different folder location.

In our example, we have chosen to consider only a change of folder location. If the fixed datasets were in the main solution and you decided to update these, then the modified main solution would have to be redeployed to MobileTogether Server and be re-downloaded to all affected client devices. This is where the server action library comes in. The (new) folder locations can be specified in the server action library, so that the main solution is not concerned with the folder location. It deals directly with the data about the respective folder, which will be forwarded to it by the server action library. The updates can be carried out in the server action library and the main solution will not need to be modified. The main solution would call an Action Group of the server action library. The Action Group would read the data of the relevant folder, store the folder data in a tree node, and return this data to the main solution. If a folder location changes, then the main solution will have the new data without having to be modified.

The screenshot below shows a control action of the main solution. The action executes a call to an Action Group named **`ScanFolder`** (which is defined in the server action library named `ServerLibrary.mtd`). The call contains a single parameter named **`$FolderToScan`**, the value of which is provided by the user of the main solution and stored there in a page source node. This user input indicates the type of collection (books, films, or music) that the user wants to scan. When the call to the Action Group is made at runtime, the Action Group will be executed in the server action library (that is outside the main solution) and the data about the scanned folder will be passed back to the main solution and stored in a solution variable named **`$FolderReadout`**. This variable, containing the data of the scanned folder, will replace the contents of the **`$XML1`** page source in the main solution (achieved with the Replace Node(s)[883] action). As a result, the data now contained in **`$XML1`** is the updated information about the scanned folder.



OnFinishEditing

OnFinishEditing 'Combo Box1'

Let variable $FolderReadout := ScanFolder

$FolderToScan := $PERSISTENT/Root/UserSelection

○ XPath ● Action Group Result ○ Subpage Result

Replace Subnodes at Target Node $XML1

Parent of source node(s) $FolderReadout

Subnode(s) *

○ as first child ● as last child

□ remove appended node(s) from their current location

ScanFolder (ServerLibrary.mtd)

Note the following points about making a call to a server action library and about the screenshot above. To see the relevant settings, you can open the tutorial file named `MainSolution.mtd` in the `ServerActionLibraries` folder of the Tutorials folder 72. In order to scan a real folder during a simulation, you will need to specify, in the server action library file (**`ServerLibrary.mtd`**), paths to real folders on your system (*see the topic Create a server action library 1555 for details*).

- The screenshot above shows the Actions dialog of the **`OnFinishEditing`** action of the main solution's only combo box control.
- The user's selection in this combo box—which can be one of the values, **`Books`**, **`Films`**, **`Music`**—will be saved in the node **`$PERSISTENT/Root/UserSelection`**.
- The server action libraries you want to access from the main solution must be added to the main solution's server action libraries in the Files Pane 259.
- Once a server action library has been added to a solution (*see previous point*), then all Action Groups of that server action library will be available automatically in Action dialogs of the solution (*see the Action Groups section in the screenshot above*).
- In the main solution, typically, a Let action 900 is used to call an Action Group of the server action library (*see screenshot above*). This is because the Let action's variable (**`$FolderReadout`** in the screenshot above) can be easily set to receive the return value of the Action Group (*see screenshot above*).
- When a call is made to an Action Group, the Action Group's parameters—defined in the server action library—are listed inside the definition of the Let action 900. The **`ScanFolder`** Action Group that is shown in the screenshot above has a parameter named **`$FolderToScan`**. You can now define the value that you want this parameter to have when the call is made to the Action Group. In our example, this parameter is the user's selection, which is made via the solution's combo box and stored in the node **`$PERSISTENT/Root/UserSelection`** as the value of the **`$FolderToScan`** variable.
- The Replace Node(s) 883 action replaces the data of the **`$XML1`** page source in the main solution with the data in the **`$FolderReadout`** variable—which is the data returned from the server action library.

# 27.1 Create a Server Action Library

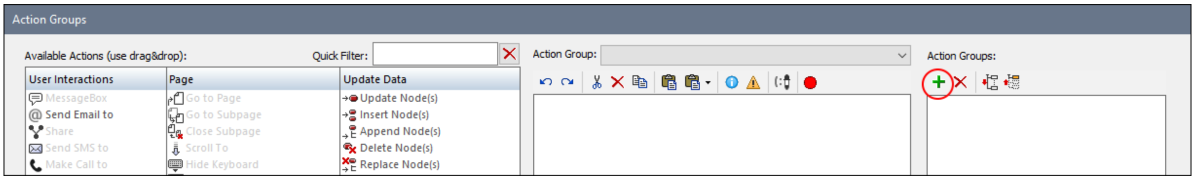
A server action library is created in the same way as you create a solution. It is essentially used for processing Action Groups 940. You can define page source trees 315 so that the server action library can use data from these trees. However, since a server action library is intended to run on the server (and so with no user interface), all controls 403 and the addition of new pages are disabled. Additionally, since a server action library can only execute server-side actions, such as sending an email from the server or updating a node in one of its page sources, it is only these kinds of actions that are enabled in a server action library; other actions are disabled.

For a higher-level description of server action libraries, see the topic server action libraries 1551

## Create a new server action library

To create a new server action library, do the following:

1. Click **File | New server action library** 1563 to open a design file for the server action library. On clicking the command, a new **server action library design** is created. The MobileTogether Designer interface will look similar to that for a solution, but a number of features will be disabled (*see first paragraph of topic*).
2. In the Main Window, click the **Action Groups** button to create an Action Group.
3. In the Action Groups dialog that appears (*screenshot below*), click the **Add a Group** button of the Action Groups pane (*button circled red in the screenshot*).



4. Create an Action Group as described in the section Action Groups 940. Note that you can define parameters for your action group. Parameters are useful if you want to pass data from the main (calling) solution to the Action Group in the server action library. Remember that the server action library does not have access to data from the calling solution. So if you want to pass data from the main solution to the Action Group in the server action library, you wil need to use parameters to do so.
5. Save the server action library with the **File | Save** 1568 command.
6. Deploy the server action library with the **File | Deploy to MobileTogether Server** 1574 command. Note that the server action library must be deployed to the same MobileTogether Server as the calling solution and the server must be the Advanced Edition. The location on the server where the server action library is saved is system-defined and will be automatically entered when you use the deploy command; it cannot be changed. (On each MobileTogether Server, all server action libraries are saved inside a single dedicated folder.)

*Important points*

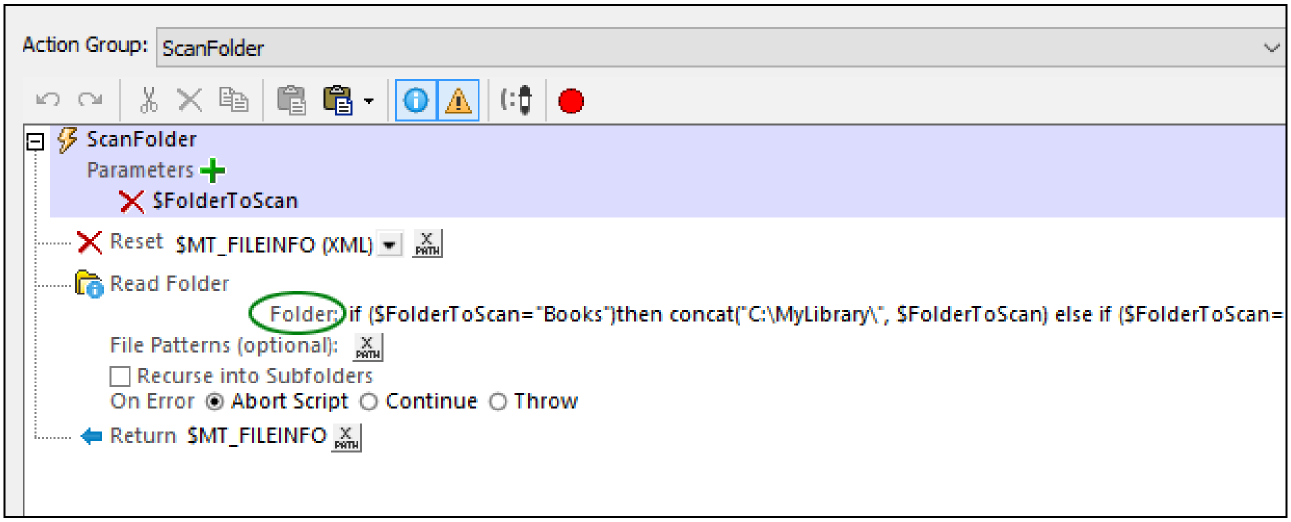
Note the following points about server action libraries:

- Deployment is to MobileTogether Server Advanced Edition.
- A server action library can call other server action libraries.
- If you modify a server action library, you must redeploy it for the changes to be accessible to calling solutions.
- How to access a server action library is described in the topic Use a server action library 1557.

## Example

The example server action library described in this topic is `ServerLibrary.mtd`, which is located in the `ServerActionLibraries` folder of the Tutorials folder[72]. You can open this server-library file and click the **Action Groups** button to go to the definition of the Action Group named **`ScanFolder`** (*screenshot below*).

In the definition of the Action Group's Read Folder[847] action, you must define paths to actual folders on your system for simulations to work. Afterward you can simulate the calling solution, `MainSolution.mtd`. For an overview of how the two example files, **`MainSolution.mtd`** and **`ServerLibrary.mtd`**, work together, see the description of the example in the topic server action libraries[1552].

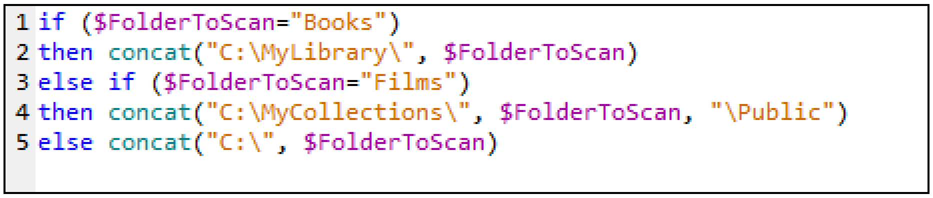


Action Group: ScanFolder

ScanFolder
Parameters
$FolderToScan
Reset $MT_FILEINFO (XML)
Read Folder
Folder: if ($FolderToScan="Books")then concat("C:\MyLibrary\", $FolderToScan) else if ($FolderToScan=
File Patterns (optional):
□ Recurse into Subfolders
On Error ● Abort Script ○ Continue ○ Throw
Return $MT_FILEINFO

Note the following points:

- The server action library contains one Action Group, named **`ScanFolder`**, the definition of which is shown in the screenshot above.
- One parameter, named **`$FolderToScan`**, has been declared for the Action Group. Its value in the Action Group will be passed in from the calling solution (see *server action libraries*[1551] and *Use a server action library*[1557])[1557].
- When the Read Folder[847] action is added to the Action Group, a page source named **`$MT_FILEINFO`** is automatically created in the server action library to hold the data obtained by the Read Folder[847] action.
- The folder to read is specified in the Folder setting of the Read Folder[847] action (*circled green in the screenshot above*). In our example, it is set via an XPath expression to read a different folder for each selection (*Books, Films,* and *Music*). Note that the conditions in the XPath expression (*see screenshot below*) use the **`$FolderToScan`** variable—which contains the user selection (*Books, Films,* or *Music*) sent from the calling solution.



```
if ($FolderToScan="Books")
then concat("C:\MyLibrary\", $FolderToScan)
else if ($FolderToScan="Films")
then concat("C:\MyCollections\", $FolderToScan, "\Public")
else concat("C:\", $FolderToScan)
```

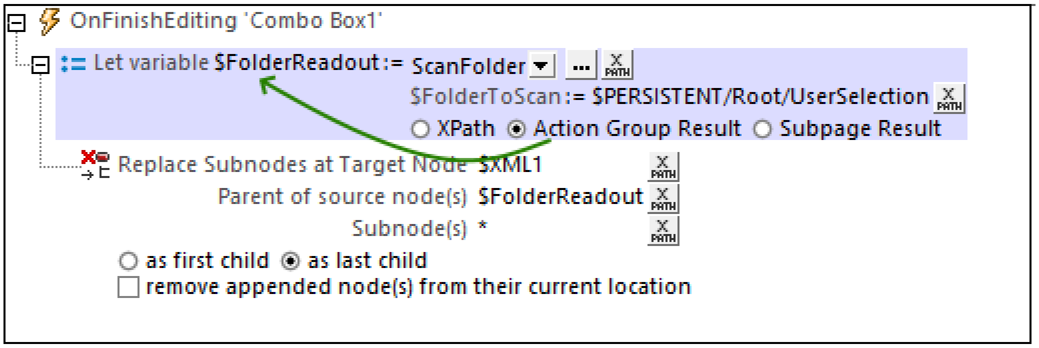
- Each folder path is built by concatenating the different parts of the respective path. It is in this XPath expression that you can modify the paths of the three folders to scan. If you want to simulate with real folders, set the three paths here to real folders on your system.
- The Reset action 802 is used to reset the **`$MT_FILEINFO`** page source before the Read Folder 847 action is executed.
- The concluding action of the Action Group, carried out by the Return action 909, is to return the **`$MT_FILEINFO`** tree as the result of the Action Group. The tree will be passed to the calling solution—to the **`$FolderReadout`** variable in the Let action 900 that calls the Action Group (*see server action libraries 1551 and Use a server action library 1557*).

**Note:** For simulations of **`MainSolution.mtd`** to work, you must define paths to actual folders on your system . These definitions are stored in the *Folder* setting of the Read Folder 847 action.

# 27.2 Use a Server Action Library

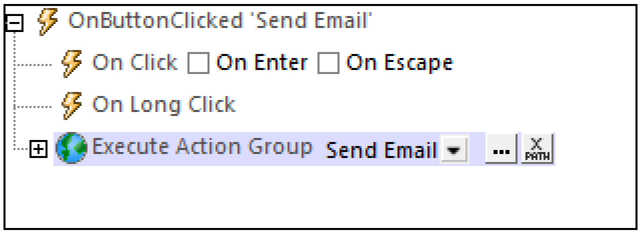
A server action library is used by calling it from a solution. Two situations arise:

- If an Action Group of a server action library is designed to return a value to the calling solution, then the call is made via a Let action[900]. This is because a Let action[900] is defined with a variable which can be set to receive the result of the Action Group (*see screenshot below*). This variable can then be used inside the calling solution. The screenshot below shows a call to the **`ScanFolder`** Action Group of a server action library via a Let action[900]. The Action Group result will be passed to the **`$FolderReadout`** variable of the calling solution.



OnFinishEditing 'Combo Box1'
Let variable $FolderReadout:= ScanFolder
$FolderToScan:= $PERSISTENT/Root/UserSelection
○ XPath ● Action Group Result ○ Subpage Result
Replace Subnodes at Target Node $XML1
Parent of source node(s) $FolderReadout
Subnode(s) *
○ as first child ● as last child
□ remove appended node(s) from their current location

- If the Action Group of the server action library is designed to carry out actions on the server (that is, independently of the calling solution and client device), then it will not have a return value to send to the calling solution. As a result, the Action Group needs only to be called at the point where it is needed; a Let action[900] is not needed. In this case, simply drag the Action Group of the server action library to the point where it should be executed in the processing of an event's actions (*see screenshot below*).

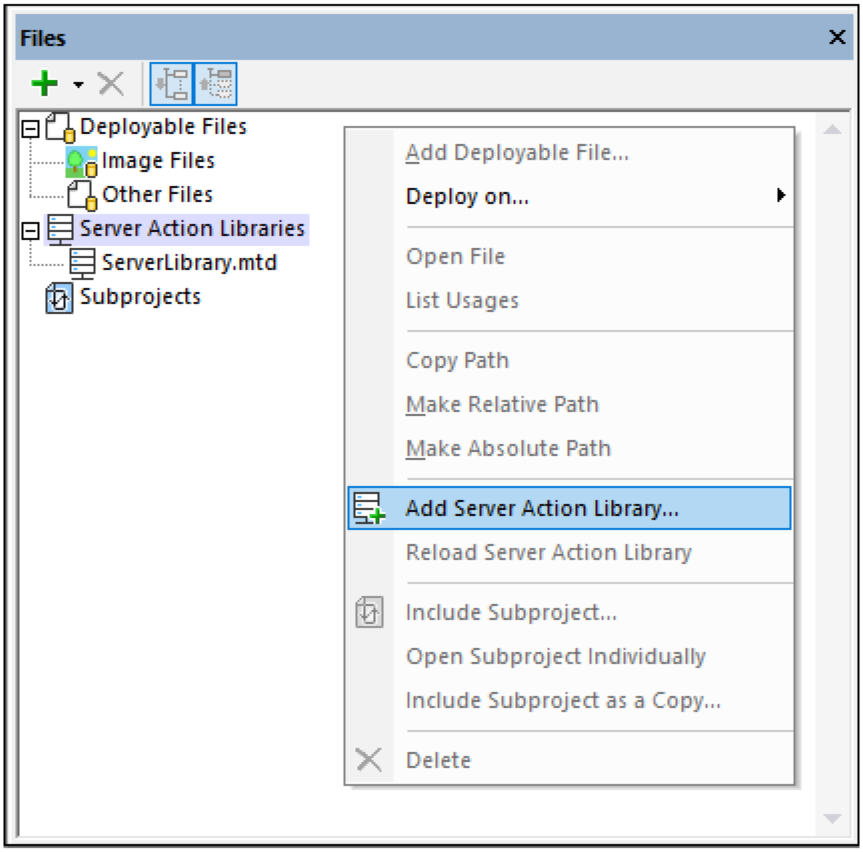


OnButtonClicked 'Send Email'
On Click □ On Enter □ On Escape
On Long Click
Execute Action Group Send Email

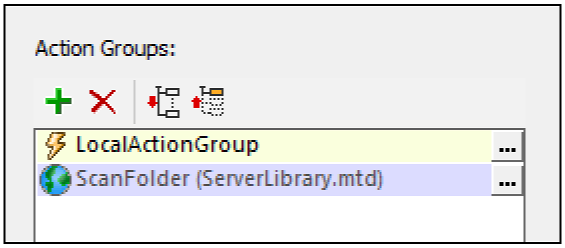
## Call the Action Group of a server action library

In a main solution, to set up a call to a server action library, do the following:

1. In the Files Pane[259], right-click the *server action libraries* item to access its context menu and select **Add server action library** (*see screenshot below*). Alternatively, use the menu command **Refactor | Add server action library**[1614].



2. In the dialog that appears, browse for the server action library you want to add, select it, and click **Open**. The server action library will be added to the list of server action libraries and its Action Groups become available for calls.
3. Go to the Actions dialog of the event for which you want to use the server action library.
4. In the Actions dialog, all Action Groups of the added server action library will be available for use (*see screenshot below*). Note that the icon of Action Groups of server action libraries is different from that of local Action Groups.



5. Drag the Action Group to the location where you want to use it (*see first screenshot at start of this topic*).
6. After saving the solution, deploy it to the same MobileTogether Server as that to which the server action library has been deployed.

*Important points*

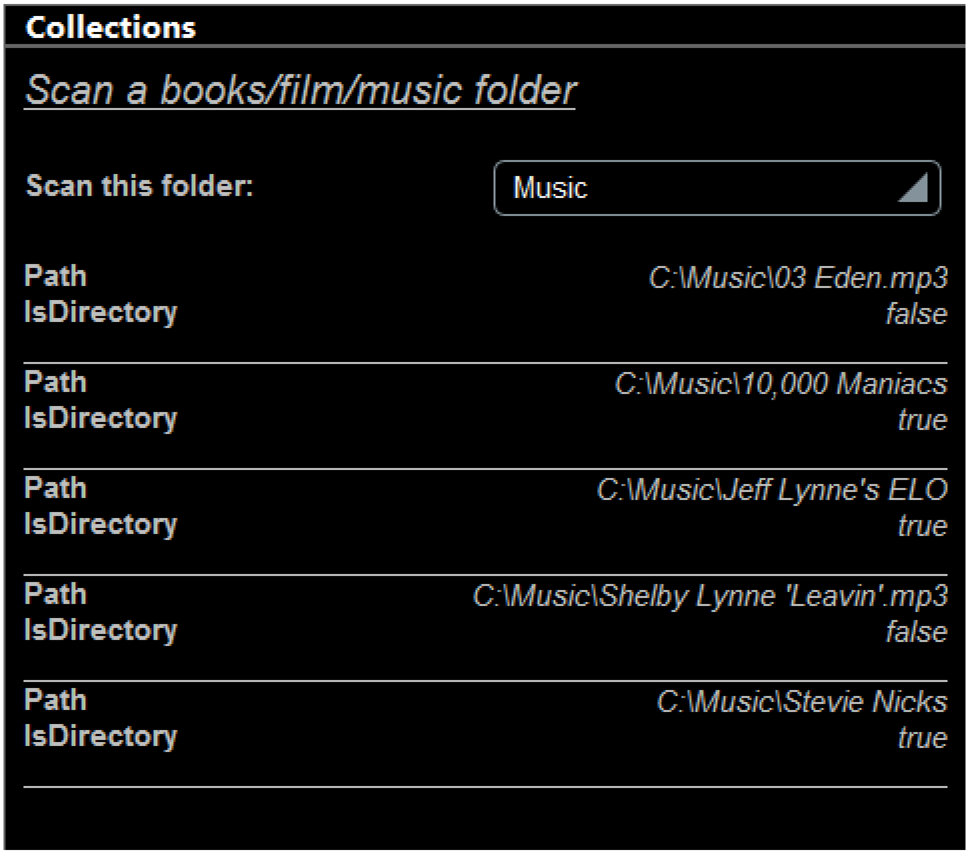
Note the following points about using Action Groups of server action libraries:

- You can add multiple server action libraries to a solution. The Action Groups of all the added server action libraries will automatically be available for use in the solution.
- If you modify a server action library in some significant way, such as by renaming a parameter, then you would need to modify the calling solution accordingly.
- If you modify a server action library, then you must reload the server action library in the Files Pane 259 of the solution to make the modifications available in the calling solution and for simulations.

## Example

In our example file, **`MainSolution.mtd`**, when the user selects a collection to scan in the *Scan this folder* combo box (*see screenshot below*), the following happens:

1. In the main (calling) solution, the collection name (*Books*, *Films*, or *Music*) is saved to the page source node **`$PERSISTENT/Root/UserSelection`** after it is selected in the solution's combo box (*see above, first screenshot of this topic, and screenshot below*).
2. The Action Group **`ScanFolder`** of the server action library **`ServerLibrary.mtd`** is called, and the user-selected collection name is passed as the value of the parameter **`$FolderToScan`** (*see above, first screenshot of this topic*).
3. The Action Group is executed and the folder corresponding to the collection name that was sent is scanned—by using the Read Folder 847 action of the server action library's Action Group (*see the topic Create a server action library 1554*).
4. The data tree that is generated as a result of the scan is returned to the calling solution, **`MainSolution.mtd`**, and stored in the calling solution's **`$FolderReadout`** variable (*see above, first screenshot of this topic*).
5. The data tree in **`$FolderReadout`** replaces the tree in **`$XML1`**.
6. The **`$XML1`** page source is displayed as a table (*screenshot below*).



Collections

Scan a books/film/music folder

Scan this folder: Music

Path C:\Music\03 Eden.mp3
IsDirectory false

Path C:\Music\10,000 Maniacs
IsDirectory true

Path C:\Music\Jeff Lynne's ELO
IsDirectory true

Path C:\Music\Shelby Lynne 'Leavin'.mp3
IsDirectory false

Path C:\Music\Stevie Nicks
IsDirectory true

# 28 Menu Commands

MobileTogether Designer menu commands are organized into the following menus:

- File 1562
- Edit 1586
- Project 1589
- Refactor 1610
- Run 1617
- Debug 1626
- Page 1631
- Table 1637
- View 1650
- Tools 1653
- Window 1681
- Help 1683

## 28.1 File

The **File** menu contains the following commands:

- New 1562
- New Service 1563
- New Server Action Library 1563
- Open 1563
- Reload 1567
- Close 1567
- Close All 1567
- Close All But Active 1567
- Save 1568
- Save As 1568
- Save Copy As 1568
- Save All 1568
- Export MobileTogether Package 1572
- Deploy to MobileTogether Server 1574
- Open from MobileTogether Server 1577
- Delete from MobileTogether Server 1578
- Generate Program Code for AppStore Apps 1580
- Send by Mail 1582
- Print 1582
- Print Preview 1583
- Print Setup 1583
- Recent Files 1585
- Exit 1585

### 28.1.1 New

⊟ *Icon*

🗋

⊟ *Shortcut*

**Ctrl+N**

⊟ *Description*

Opens a new document tab in the main window and loads an empty MobileTogether Design file into this tab. The document is stored temporarily in memory. It must be saved to disk with the `.mtd` extension if you want to keep it.

## 28.1.2 New Service

⊟ *Icon*

⊟ *Description*

Opens a new document tab in the main window and loads an empty MobileTogether Server Service file into this tab. The document is stored temporarily in memory. It must be saved to disk with the `.mtd` extension if you want to keep it. For a description of the MobileTogether Server Service features, see the section Server Services 1543.

## 28.1.3 New Server Action Library

⊟ *Icon*

⊟ *Description*

On clicking the New Server Action Library command, a new tab is opened and a new **server action library design** is created. The MobileTogether Designer interface will look similar to that for a solution, but a number of features will be disabled. You can define page source trees 315 so that the server action library can use data from these trees. However, since a server action library is intended to run on the server (and so with no user interface), all controls 403 and the addition of new pages are disabled. Additionally, since a server action library can only execute server-side actions, such as sending an email from the server or updating a node in one of its page sources, it is only these kinds of actions that are enabled in a server action library; other actions are disabled. See the topic Server Action Libraries 1551 for a detailed description.

## 28.1.4 Open

▼ Open

⊟ *Icon*

⊟ *Shortcut*

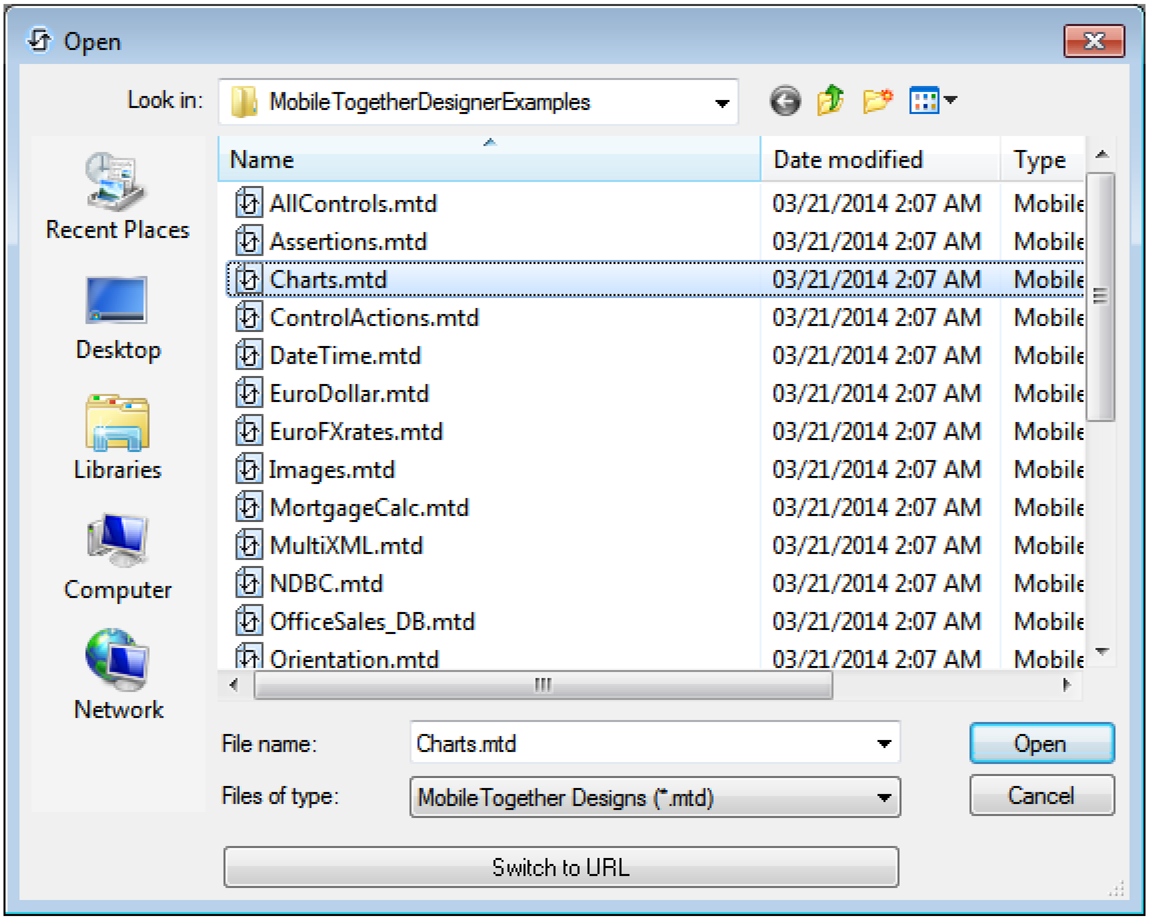
**Ctrl+O**

⊟ *Description*

Pops up the Open dialog, in which you can select the MobileTogether Design file (`.mtd` file) to open. The MTD file is opened in a new tab of the main window.

▼ Selecting and saving files via URLs

In several File Open and File Save dialogs, you can choose to select the required file or save a file via a URL (*see screenshot below*). Click **Switch to URL** to go to the selection process.



Look in: MobileTogetherDesignerExamples

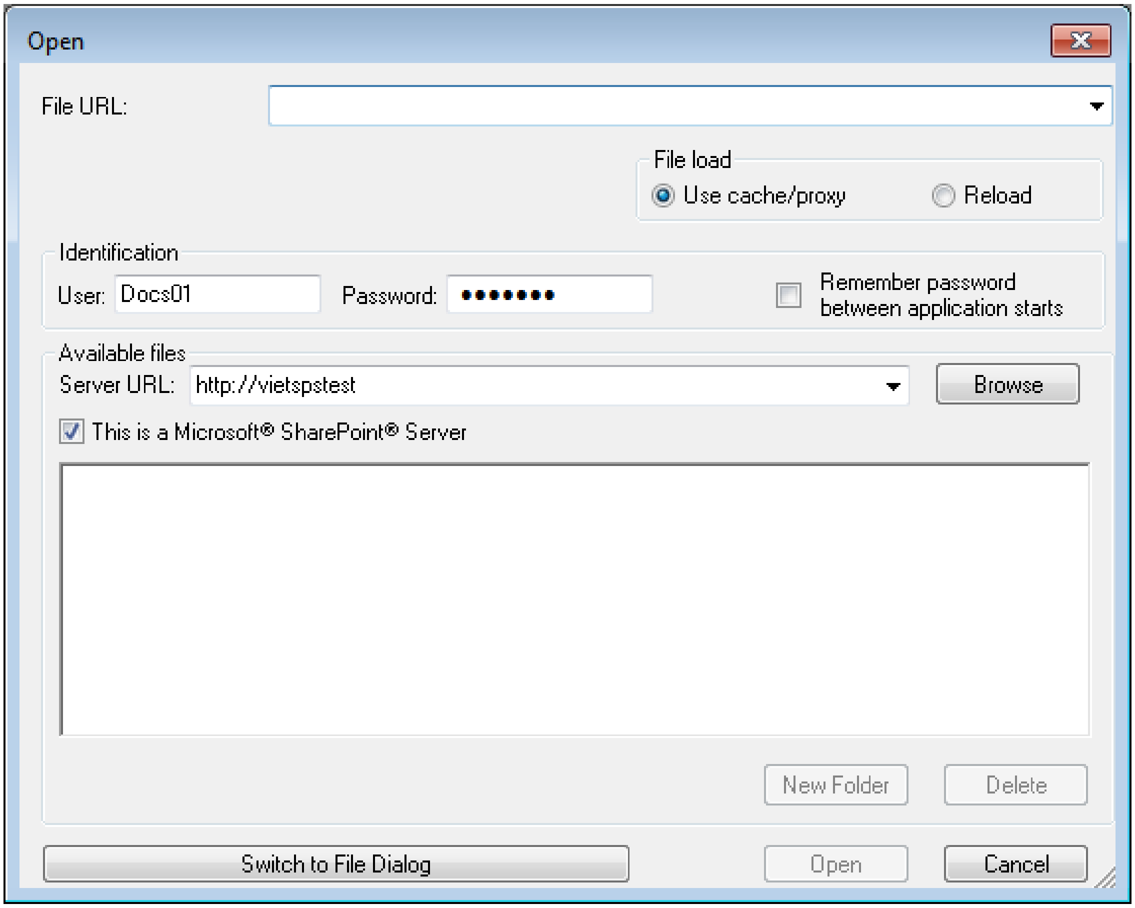
| Name | Date modified | Type |
|---|---|---|
| AllControls.mtd | 03/21/2014 2:07 AM | Mobile |
| Assertions.mtd | 03/21/2014 2:07 AM | Mobile |
| Charts.mtd | 03/21/2014 2:07 AM | Mobile |
| ControlActions.mtd | 03/21/2014 2:07 AM | Mobile |
| DateTime.mtd | 03/21/2014 2:07 AM | Mobile |
| EuroDollar.mtd | 03/21/2014 2:07 AM | Mobile |
| EuroFXrates.mtd | 03/21/2014 2:07 AM | Mobile |
| Images.mtd | 03/21/2014 2:07 AM | Mobile |
| MortgageCalc.mtd | 03/21/2014 2:07 AM | Mobile |
| MultiXML.mtd | 03/21/2014 2:07 AM | Mobile |
| NDBC.mtd | 03/21/2014 2:07 AM | Mobile |
| OfficeSales_DB.mtd | 03/21/2014 2:07 AM | Mobile |
| Orientation.mtd | 03/21/2014 2:07 AM | Mobile |

File name: Charts.mtd

Files of type: MobileTogether Designs (*.mtd)

To select a file via a URL (either for opening or saving), do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open or Save dialog (*the screenshot below shows the Open dialog*).



File URL:

File load
● Use cache/proxy ○ Reload

Identification
User: Docs01 Password: ••••••• □ Remember password between application starts

Available files
Server URL: http://vietspstest
☑ This is a Microsoft® SharePoint® Server

2. Enter the URL you want to access in the *Server URL* field (*screenshot above*). If the server is a Microsoft® SharePoint® Server, check the *Microsoft® SharePoint® Server* check box. See the Microsoft® SharePoint® Server Notes below for further information about working with files on this type of server.
3. If the server is password protected, enter your User-ID and password in the *User* and *Password* fields.
4. Click **Browse** to view and navigate the directory structure of the server.
5. In the folder tree, browse for the file you want to load and click it. The file URL appears in the File URL field. The **Open** or **Save** button only becomes active at this point.
6. Click **Open** to load the file or **Save** to save it.

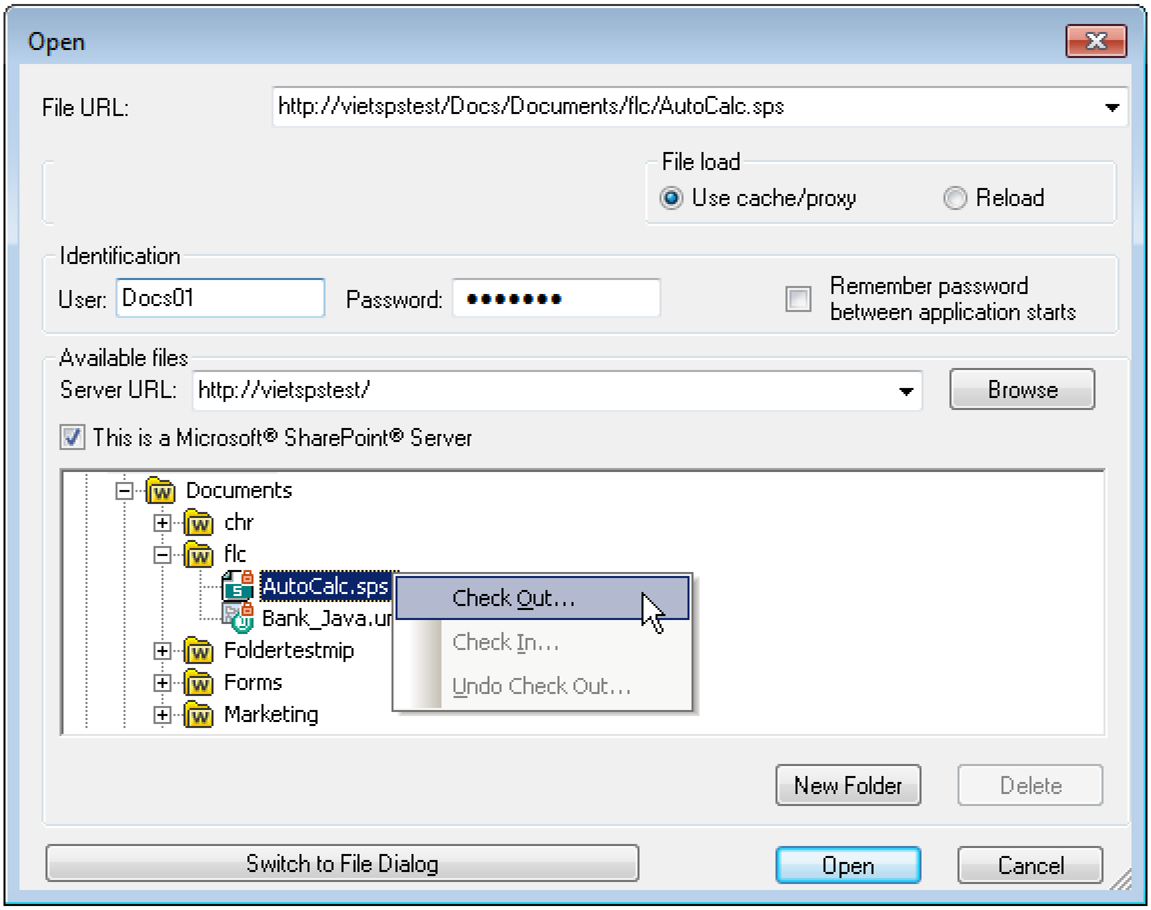
*Note the following*:

- The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.
- To give you more control over the loading process when opening a file, you can choose to load the file through the local cache or a proxy server (which considerably speeds up the process if the file has been loaded before). Alternatively, you may want to reload the file if you are working, say, with an electronic publishing or database system; select the **Reload** option in this case.

### Microsoft® SharePoint® Server Notes

Note the following points about files on Microsoft® SharePoint® Servers:

- In the directory structure that appears in the Available Files pane (*screenshot below*), file icons have symbols that indicate the check-in/check-out status of files.



Open

File URL: http://vietspstest/Docs/Documents/flc/AutoCalc.sps

File load: ● Use cache/proxy ○ Reload

Identification

User: Docs01 Password: ●●●●●●●

□ Remember password between application starts

Available files

Server URL: http://vietspstest/

☑ This is a Microsoft® SharePoint® Server

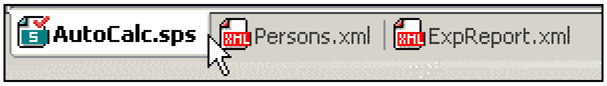
Right-clicking a file pops up a context menu containing commands available for that file (*screenshot above*).

- The various file icons are shown below:

| | |
|---|---|
| XSD | Checked in. Available for check-out. |
| XML | Checked out by another user. Not available for check-out. |
| S | Checked out locally. Can be edited and checked-in. |

- After you check out a file, you can edit it in your Altova application and save it using **File | Save (Ctrl+S)**.

- You can check-in the edited file via the context menu in the Open URL dialog (*see screenshot above*), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (*screenshot below*).



- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft® SharePoint® Server will be: **Check In** and **Undo Check Out**.

## 28.1.5 Reload

⊟ *Icon*

⊟ *Description*

Reloads any open documents that have modified outside MobileTogether Designer. If one or more documents is modified outside MobileTogether Designer, a prompt appears asking whether you wish to reload the modified document/s. If you choose to reload, then any changes you may have made to the file since the last time it was saved will be lost.

## 28.1.6 Close, Close All, Close All But Active

▼ Close

⊟ *Description*

Closes the active document window. If the file was modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

▼ Close All

⊟ *Description*

Closes all open document windows. If any document has been modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

▼ Close All But Active

⊟ *Description*

Closes all open document windows except the active document window. If any document has been modified (indicated by an asterisk * after the file name in the title bar), you will be asked if you wish to save the file first.

### 28.1.7 Save, Save As, Save Copy As, Save All

▼ Save

⊟ *Icon*

💾

⊟ *Shortcut*

**Ctrl+S**

⊟ *Description*

Saves the contents of the active document to the file from which it has been opened.

▼ Save As

⊟ *Description*

Pops up the Save As dialog box, in which you enter the name and location of the file you wish to save the active document as. The newly saved document replaces the original document in the active tab of the main window.

▼ Save Copy As

⊟ *Description*

Pops up the Save Copy As dialog box, in which you enter the name and location of the file you wish to save the active document as. The saved document will be a copy of the active document. The newly saved document will not be opened in MobileTogether Designer. The original document remains active in the main window.

▼ Save All

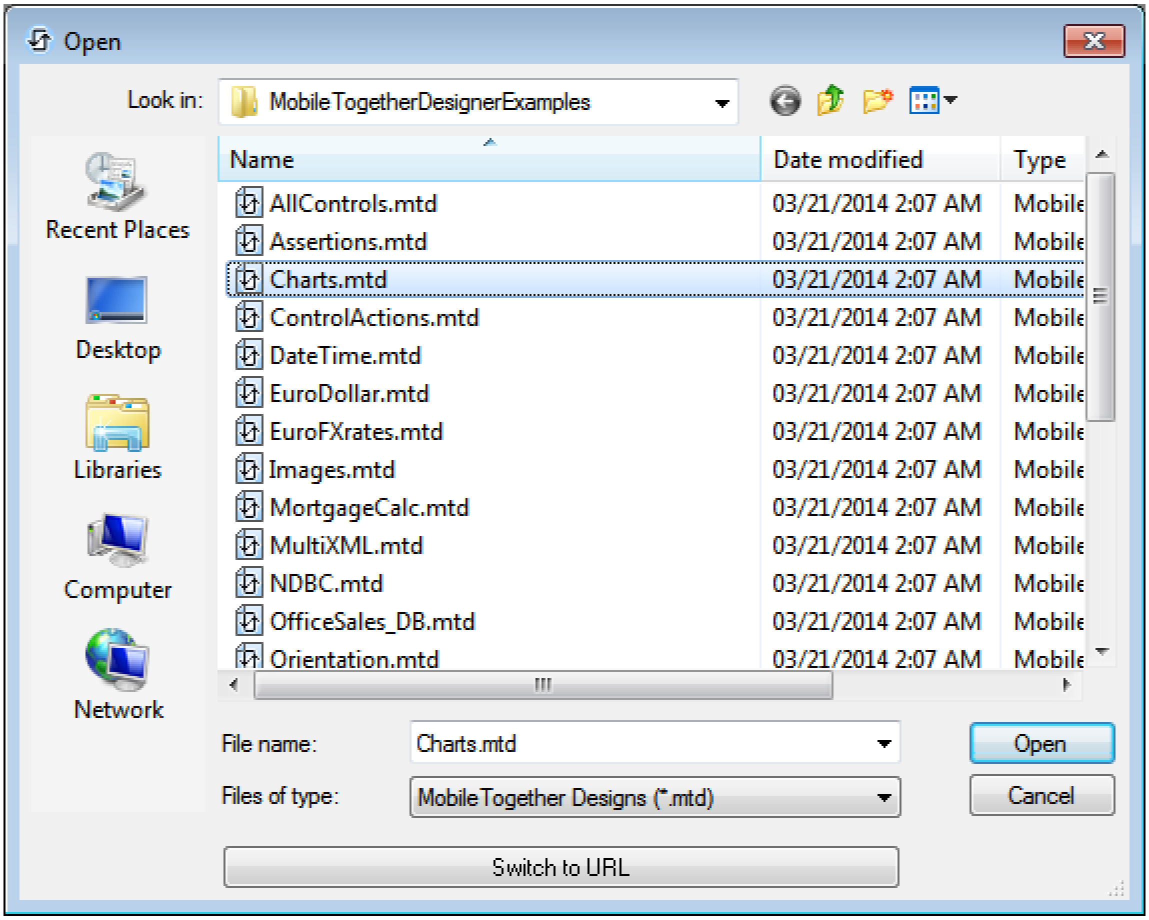
⊟ *Icon*

⊟ *Description*

Saves all modifications that have been made to any open documents. The command is useful if you edit multiple documents simultaneously. If a document has not been saved before (for example, after being newly created), the Save As dialog box is presented for that document.

▼ Selecting and saving files via URLs

In several File Open and File Save dialogs, you can choose to select the required file or save a file via a URL (*see screenshot below*). Click **Switch to URL** to go to the selection process.



Look in: MobileTogetherDesignerExamples

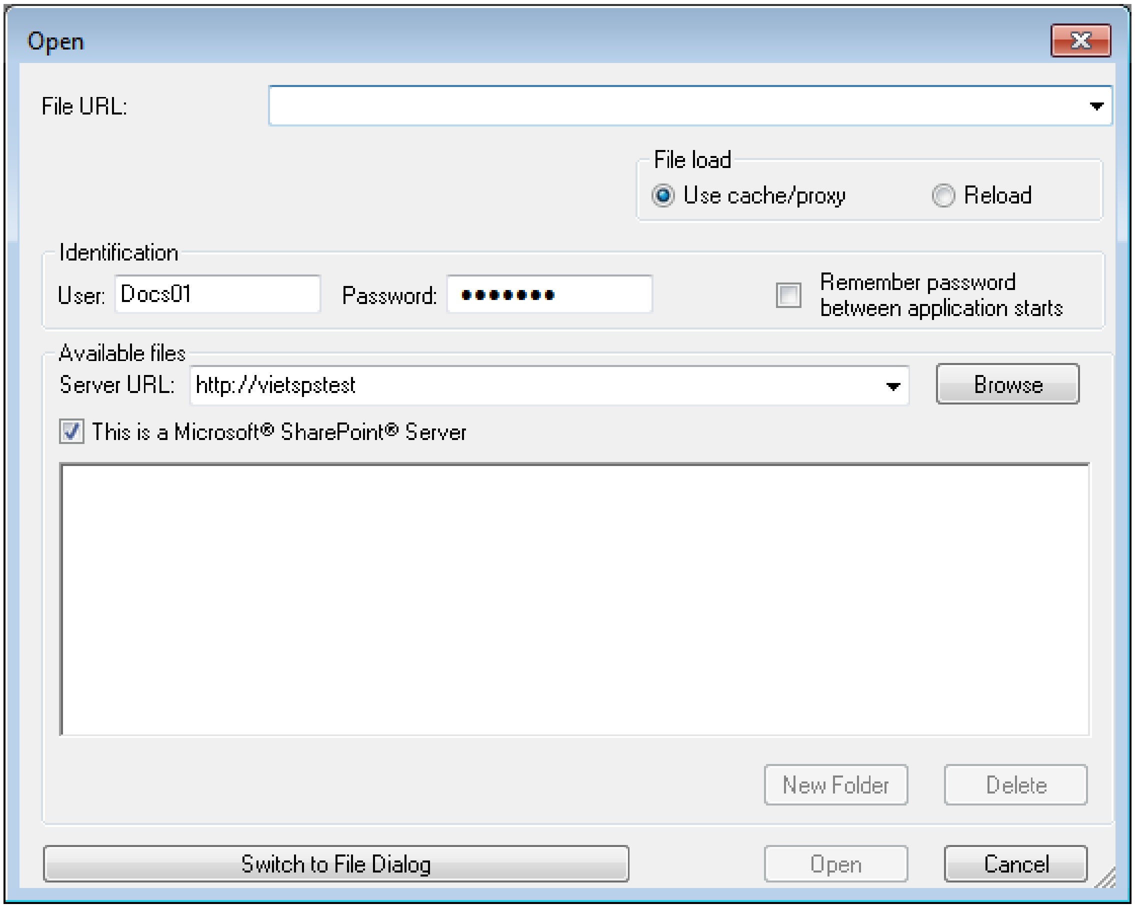
| Name | Date modified | Type |
|---|---|---|
| AllControls.mtd | 03/21/2014 2:07 AM | Mobile |
| Assertions.mtd | 03/21/2014 2:07 AM | Mobile |
| Charts.mtd | 03/21/2014 2:07 AM | Mobile |
| ControlActions.mtd | 03/21/2014 2:07 AM | Mobile |
| DateTime.mtd | 03/21/2014 2:07 AM | Mobile |
| EuroDollar.mtd | 03/21/2014 2:07 AM | Mobile |
| EuroFXrates.mtd | 03/21/2014 2:07 AM | Mobile |
| Images.mtd | 03/21/2014 2:07 AM | Mobile |
| MortgageCalc.mtd | 03/21/2014 2:07 AM | Mobile |
| MultiXML.mtd | 03/21/2014 2:07 AM | Mobile |
| NDBC.mtd | 03/21/2014 2:07 AM | Mobile |
| OfficeSales_DB.mtd | 03/21/2014 2:07 AM | Mobile |
| Orientation.mtd | 03/21/2014 2:07 AM | Mobile |

File name: Charts.mtd

Files of type: MobileTogether Designs (*.mtd)

To select a file via a URL (either for opening or saving), do the following:

1. Click the **Switch to URL** command. This switches to the URL mode of the Open or Save dialog (*the screenshot below shows the Open dialog*).



Open

File URL:

File load

● Use cache/proxy ○ Reload

Identification

User: Docs01 Password: •••••••

□ Remember password between application starts

Available files

Server URL: http://vietspstest

☑ This is a Microsoft® SharePoint® Server

2. Enter the URL you want to access in the *Server URL* field (*screenshot above*). If the server is a Microsoft® SharePoint® Server, check the *Microsoft® SharePoint® Server* check box. See the Microsoft® SharePoint® Server Notes below for further information about working with files on this type of server.
3. If the server is password protected, enter your User-ID and password in the *User* and *Password* fields.
4. Click **Browse** to view and navigate the directory structure of the server.
5. In the folder tree, browse for the file you want to load and click it. The file URL appears in the File URL field. The **Open** or **Save** button only becomes active at this point.
6. Click **Open** to load the file or **Save** to save it.

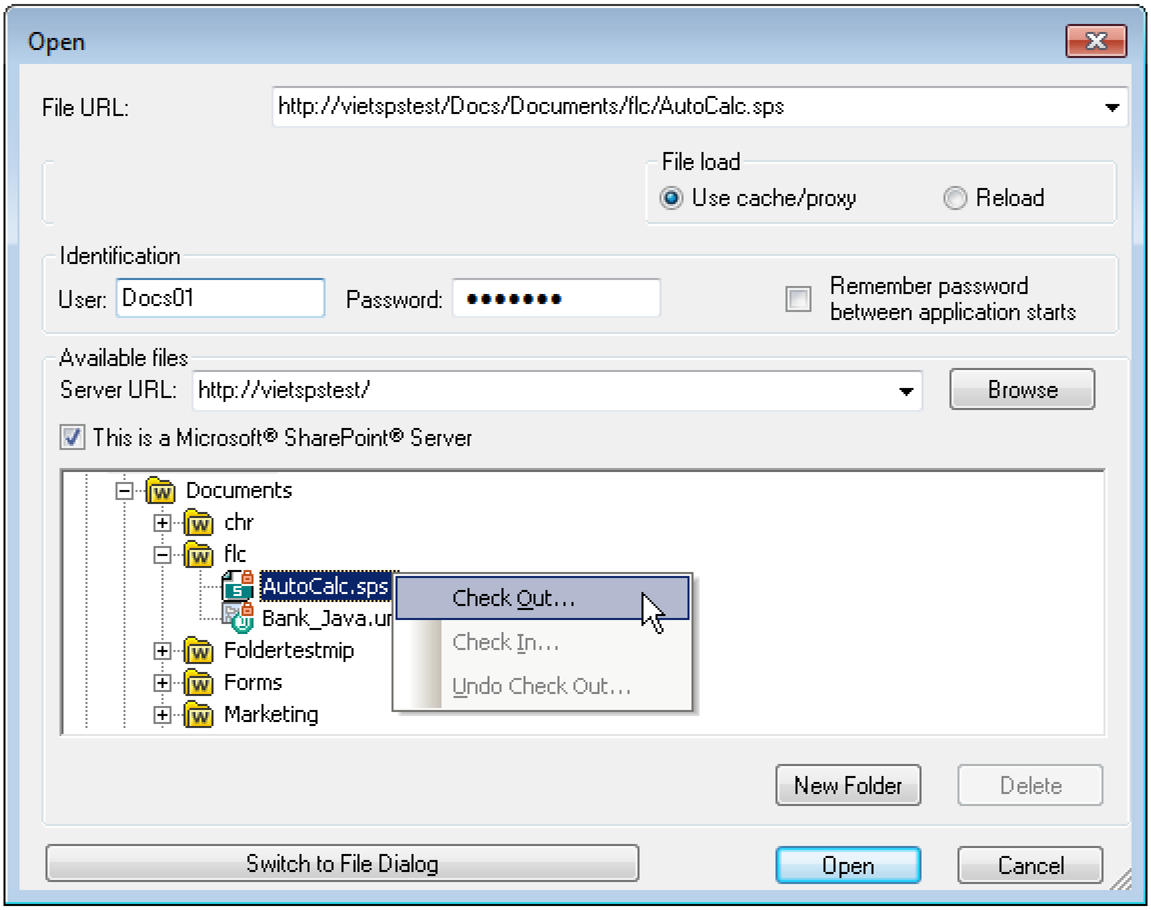
*Note the following:*

- The Browse function is only available on servers which support WebDAV and on Microsoft SharePoint Servers. The supported protocols are FTP, HTTP, and HTTPS.
- To give you more control over the loading process when opening a file, you can choose to load the file through the local cache or a proxy server (which considerably speeds up the process if the file has been loaded before). Alternatively, you may want to reload the file if you are working, say, with an electronic publishing or database system; select the **Reload** option in this case.

▼ Microsoft® SharePoint® Server Notes

Note the following points about files on Microsoft® SharePoint® Servers:

- In the directory structure that appears in the Available Files pane (*screenshot below*), file icons have symbols that indicate the check-in/check-out status of files.



Open

File URL: http://vietspstest/Docs/Documents/flc/AutoCalc.sps

File load
● Use cache/proxy ○ Reload

Identification
User: Docs01 Password: ••••••• □ Remember password between application starts

Available files
Server URL: http://vietspstest/ Browse
☑ This is a Microsoft® SharePoint® Server

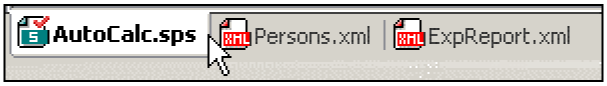
Right-clicking a file pops up a context menu containing commands available for that file (*screenshot above*).

- The various file icons are shown below:

| | |
|---|---|
| XSD | Checked in. Available for check-out. |
| XML | Checked out by another user. Not available for check-out. |
| S | Checked out locally. Can be edited and checked-in. |

- After you check out a file, you can edit it in your Altova application and save it using **File | Save (Ctrl+S)**.

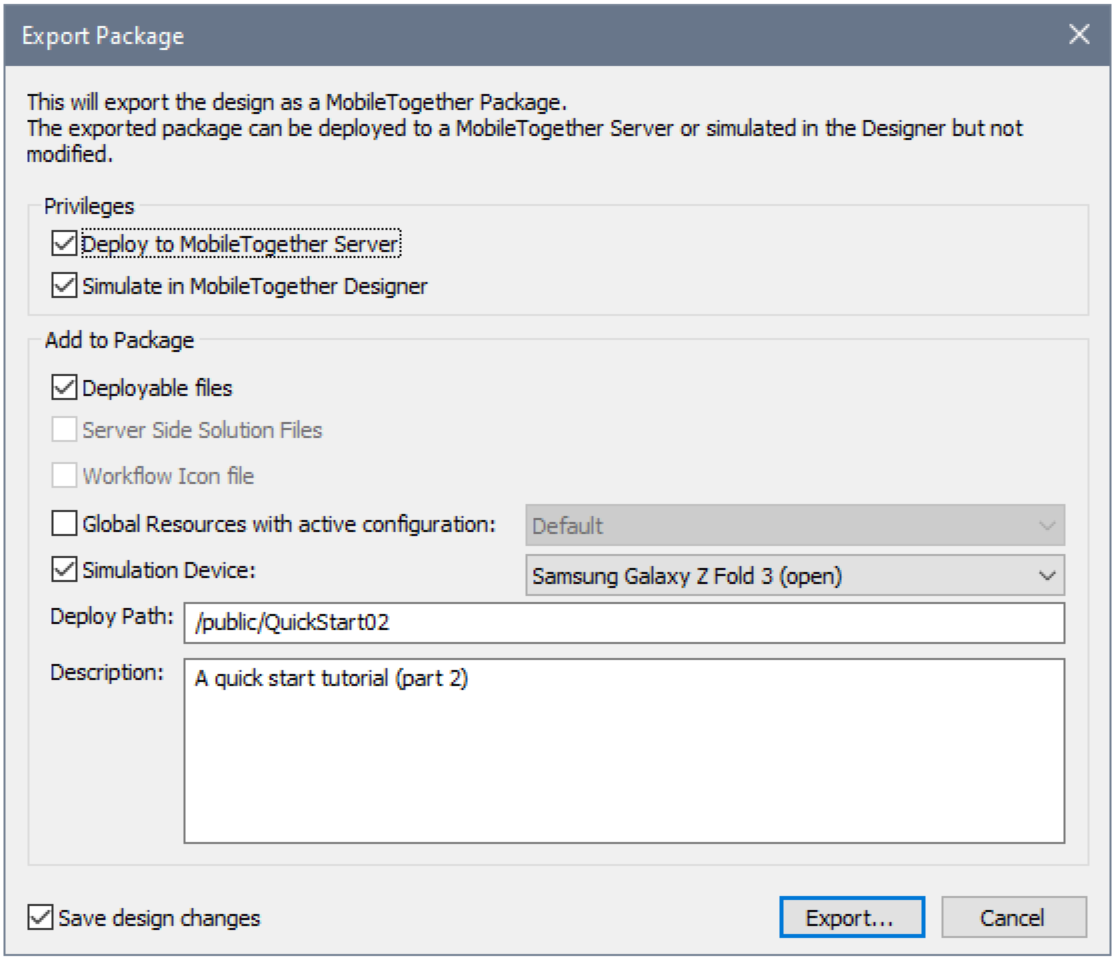
- You can check-in the edited file via the context menu in the Open URL dialog (*see screenshot above*), or via the context menu that pops up when you right-click the file tab in the Main Window of your application (*screenshot below*).



- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in one Altova application, you cannot check it out in another Altova application. The file is considered to be already checked out to you. The available commands at this point in any Altova application supporting Microsoft® SharePoint® Server will be: **Check In** and **Undo Check Out**.

## 28.1.8 Export MobileTogether Package

Opens the Export Package dialog *(screenshot below)* for creating a MobileTogether package file (`.mtp` file) from the active MobileTogether design. The MobileTogether package is a zipped archive that contains the design file in an encrypted form and, optionally, additional deployable resource files that are used by the design (such as images and CSS stylesheets). See MobileTogether Packages 295 for more information.



Export Package

This will export the design as a MobileTogether Package.
The exported package can be deployed to a MobileTogether Server or simulated in the Designer but not modified.

Privileges

☑ Deploy to MobileTogether Server

☑ Simulate in MobileTogether Designer

Add to Package

☑ Deployable files

□ Server Side Solution Files

□ Workflow Icon file

□ Global Resources with active configuration: Default

☑ Simulation Device: Samsung Galaxy Z Fold 3 (open)

Deploy Path: /public/QuickStart02

Description: A quick start tutorial (part 2)

☑ Save design changes

In the Export Package dialog *(screenshot above)*, select the resources you wish to include in the MobileTogether package:

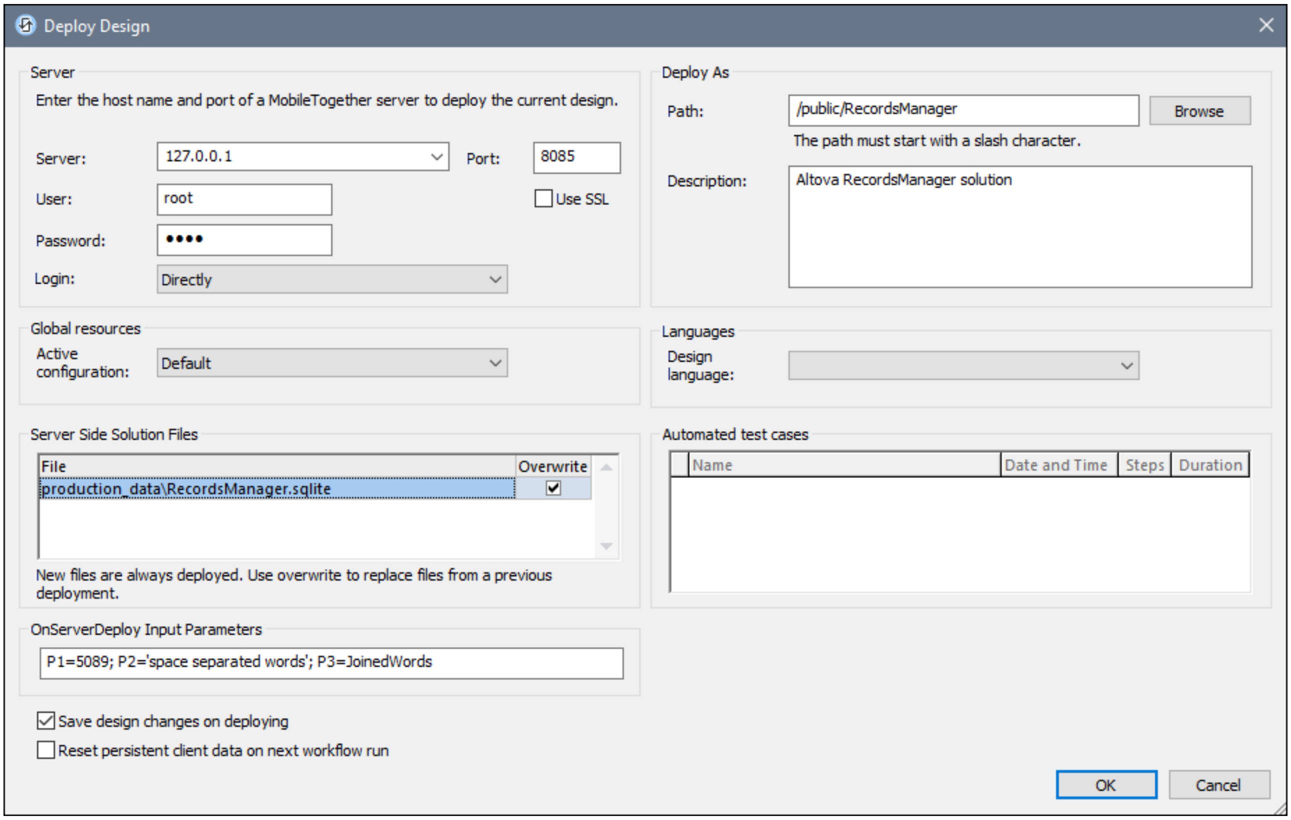
- *Privileges:* Will the package be used for deployment, for simulations, or both. See MobileTogether Packages 295 for more information.
- *Deployable files:* These resources will be exported with the design. Note that deployable files are saved in MobileTogether's database on the server and cannot be modified. Typically these are image files or files containing unchanging data. Files that cannot be deployed would be data files that are modified during a solution run, and are, therefore, not stored in the MobileTogether database on the server. Such non-deployable files must be saved separately to the server. See Deploying the Project 291 for more information.
- *Server Side Solution Files:* These files will also be saved in the package. When the package is deployed to the server 1574, the server side solution files will also be available for deployment.
- *Workflow Icon file:* This option is enabled only if a Workflow Icon file has been set in the project's properties 296.
- *Global resources with active configuration:* Select the active configuration you want to use for global resources. The corresponding global resources will be added to the package.
- *Simulation Device:* Specify the device to be used for simulations.
- *Deploy Path:* This is the path to the solution on the server when the package is deployed.

**Note:** Test cases 1399 that have been saved with the design are not exported to the package.

Click **Export** when ready. In the dialog that then appears, enter the location where you want to save the package file and click **Save**. The design and the selected resources will be saved in a MobileTogether package file that has a `.mtp` file extension. See MobileTogether Packages 295 for information about using package files.

## 28.1.9 Deploy to MobileTogether Server

Opens the Deploy Design dialog, in which you specify the deployment details of the currently active design or package. The deployment settings are described below. On clicking **OK**, the design or package is deployed to MobileTogether Server as a solution.



Deploy Design

Server

Enter the host name and port of a MobileTogether server to deploy the current design.

Server: 127.0.0.1 Port: 8085

User: root □ Use SSL

Password: ••••

Login: Directly

Global resources

Active configuration: Default

Server Side Solution Files

| File | Overwrite |
|---|---|
| production_data\RecordsManager.sqlite | ☑ |

New files are always deployed. Use overwrite to replace files from a previous deployment.

OnServerDeploy Input Parameters

P1=5089; P2='space separated words'; P3=JoinedWords

☑ Save design changes on deploying

□ Reset persistent client data on next workflow run

Deploy As

Path: /public/RecordsManager

The path must start with a slash character.

Description: Altova RecordsManager solution

Languages

Design language:

Automated test cases

| Name | Date and Time | Steps | Duration |
|---|---|---|---|

**Note:** When a project is deployed, the actions of the **`OnServerDeployment`** event are executed. (You can define the actions of this event via the *More Project Settings* 296 property of the project.) During the action-handling, the server is locked and clients will not be able to connect to it. If locking is not possible within 10 seconds, a deployment error is reported.

## Settings

*Server name and port*
The port refers to the MobileTogether Server administrator port and must match the Administrator Ports setting of MobileTogether Server. If you use SSL, make sure that you use the secure port of MobileTogether Server.

*User name and password*
The user name and password of a user that has been given the MobileTogether Server privilege *Save workflow from designer*. MobileTogether Server users and their privileges are specified in the Users and Roles settings of MobileTogether Server.

*Login*
Specify whether login is direct or as a domain user. (If login is as a domain user, then the domain-specific user name and password can be used.) From the options in the combo box, select *Directly* or the domain you wish to use. Only those domains are displayed in the combo box that have been configured on the server for active directory login. For more information about configuring the server for active directory login, see the MobileTogether Server user manual.

*Deployment path and solution description*
The path and name of the deployed solution, and the description of the solution that will appear on the server.

*Active configuration of global resources*
Select the active configuration from the available configurations in the dropdown list of the combo box. The available configurations are those that are defined in the Global Resources Definitions file and are automatically obtained from there. See the section Altova Global Resources 1334 for details.

*Design language*
Select the language that will be used by default when the solution is opened on a client. The dropdown list of the combo box displays the languages that have been defined in the design's Localization dialog 1600. Note that, after deploying the solution to MobileTogether Server, you can change the starting language of the solution in the *Workflows* tab of MobileTogether Server.

- *Auto* specifies that the language of the client will be used.
- *Default* selects the design's default language, which is the language in which the solution is designed.
- The other items in the dropdown list are the names of the localizations (the languages into which the design's default language is translated).

*Server side solution files*
Lists all the server side solution files of the active design, for each of which there is an option to overwrite the file of the same name on the server. These files will be deployed to the server, to a location relative to the *Server Side Solution's Working Directory*.

If a file in the list has not yet been deployed, then it will be deployed (even if the *Overwrite* check box has not been ticked). If a file already exists on the server, then the *Overwrite* option can be used to overwrite the existing file. If the files cannot be written (say, for example, because another client of the solution is currently using the server), then the deployment attempt waits 10 seconds for the server to be available. If there is no success within 10 seconds, then deployment fails and an error message to the effect is returned.

If both MobileTogether Designer and MobileTogether Server support the WebSocket deployment protocol, then a Deploying... dialog is displayed after you click **OK** in the Deploy Design dialog. The Deploying dialog contains a progress bar and displays the name of the file that is being currently uploaded to the server. The dialog also contains a **Cancel** button that enables you to cancel the deployment.

The deployment of large files (100+ MB) is supported from version 8.1 onwards. Older versions of MobileTogether Designer do not support the WebSocket protocol. Consequently, the Deploying... dialog is not displayed in these versions.

**Note:** If a server side solution file is to be deployed, then **all clients** of **all the solutions on the server** will be locked for the duration of the deployment. All clients and solutions will automatically be unlocked when deployment has finished. Clients that used the solution for which new files have been deployed must be restarted.

**Note:** Before attempting an overwrite, a backup file is created in the same folder as the file to be overwritten. The backup file has the same name as the target file plus a date-time extension. If the overwrite is successful, then the backup is deleted. If the overwrite fails, then the original file is restored from the backup and the backup is deleted. If the restoration fails, then the backup is not deleted.

*OnServerDeployment Input Parameters*
This entry field becomes available only if at least one action has been defined for the **`OnServerDeployment`** event (which is accessed via the More Project Settings project property 296). The input parameters that you enter will be passed to the **`$MT_InputParameters`** 1300 global variable at the time of deployment, and the parameter values can then be retrieved from this variable and used in **`OnServerDeployment`** actions.

Input parameters are entered as name–value pairs that are separated by semicolons. Strings that contain spaces must be enclosed with double quotes or single quotes.

```
    Param-1=5089; MyParam-2="space separated words"; SomeParam-3=JoinedWords
```

Note that when these values are passed to **`$MT_InputParameters`** 1300, they are stored there as strings—number values, as well. If in the project settings 296 you have specified that the values in **`$MT_InputParameters`** 1300 are stored as a sequence of values (and not as the default map), then the resulting sequence of values will be sorted alphabetically on the key names of the parameters and each value will then have to be accessed via its respective index position in the sequence, for example **`$MT_InputParameters[1]`**.

For example, you could (i) reset a user password by updating a node with a new password that is provided by an input parameter and (ii) send the user an email with the new password, with this password in the email being provided by the same input parameter.

**Note:** If a parameter value is stored in a node of the **`$PERSISTENT`** page source, then you would need to reset this node (at a suitable time) for the eventuality that on a subsequent server deployment no parameter value is passed to the solution.

*Automated test cases*
Displays the recorded base test runs 1399 (the test cases) of the design. Select the test cases that you want to deploy. Use the **Ctrl** and **Shift** keys to select multiple test cases.

*Save design changes on deploying*
The project file is saved before deploying, so that the latest changes to the design are included.

*Reset persistent client data on next workflow run*
Resets persistent client data when the solution is run the next time.

⊟ Also see

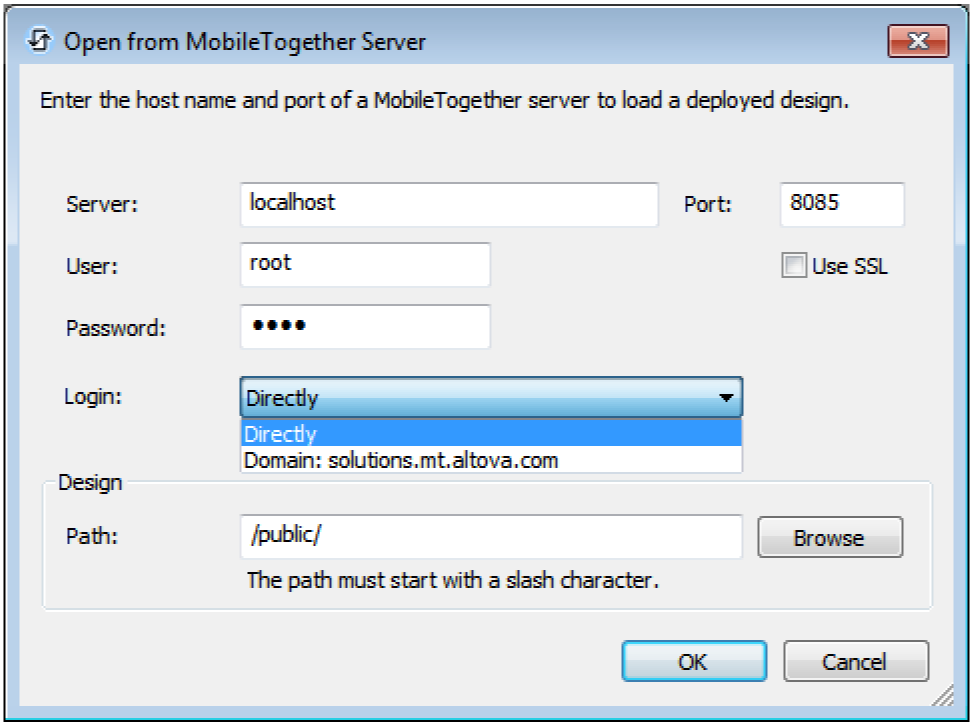
Location of Project Files 289
Files Pane 259
Deploying the Project 291
Data Storage on Servers 312.

## 28.1.10 Open from MobileTogether Server

⊟ *Icon*

⊟ *Description*

Opens a MobileTogether design file that has been deployed to MobileTogether Server server from its location on MobileTogether Server. Use the **Browse** button to select the file on the server that you want to open.



Open from MobileTogether Server

Enter the host name and port of a MobileTogether server to load a deployed design.

Server: localhost Port: 8085

User: root □ Use SSL

Password: ••••

Login: Directly

Directly
Domain: solutions.mt.altova.com

Design

Path: /public/

The path must start with a slash character.

The following details are required for opening a design from MobileTogether Server:

- *Server name and port:* The port refers to the MobileTogether Server administrator port and must match the *Administrator Port setting* of MobileTogether Server. If you use SSL, make sure that you use the secure port of MobileTogether Server.

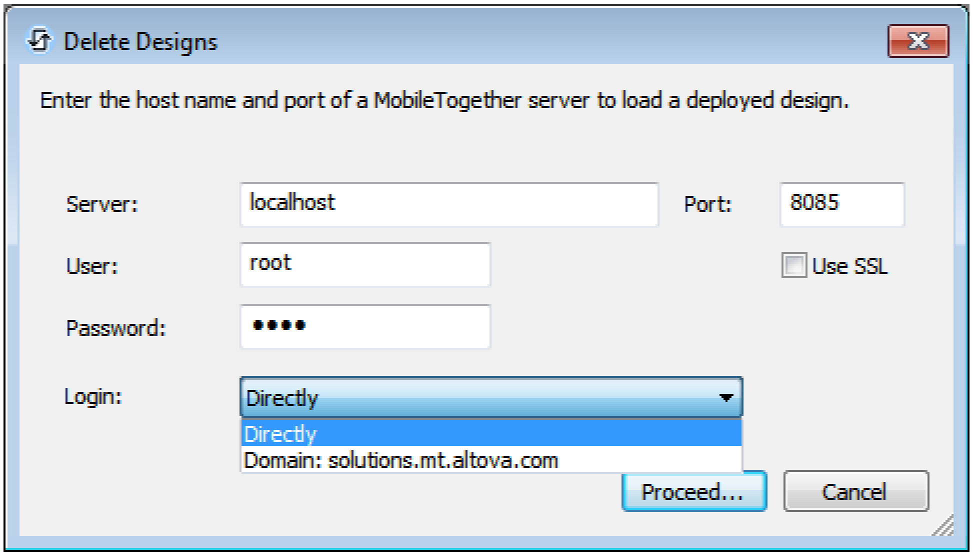
- *User name and password:* The user name and password of a user that has been given the privilege *Open workflow from designer*. MobileTogether Server users and their privileges are specified in the *Users and Roles settings* of MobileTogether Server.
- *Login:* Specify whether login is direct or as a domain user. (If login is as a domain user, then the domain-specific user name and password can be used.) From the options in the combo box, select *Directly* or the domain you wish to use. Only those domains are displayed in the combo box that have been configured on the server for active directory login. *For more information about configuring the server for active directory login, see the MobileTogether Server user manual.*
- *Path and name of solution (design):* The path and name of the deployed solution. Click **Browse** to browse through all deployed solutions on MobileTogether Server.

## 28.1.11 Delete from MobileTogether Server

*Icon*

*Description*

Deletes a previously deployed design file from MobileTogether Server. Use the **Proceed** button to select the file(s) on the server that you want to delete.



Delete Designs

Enter the host name and port of a MobileTogether server to load a deployed design.

Server: localhost

Port: 8085

User: root

□ Use SSL

Password: ••••

Login: Directly

Directly

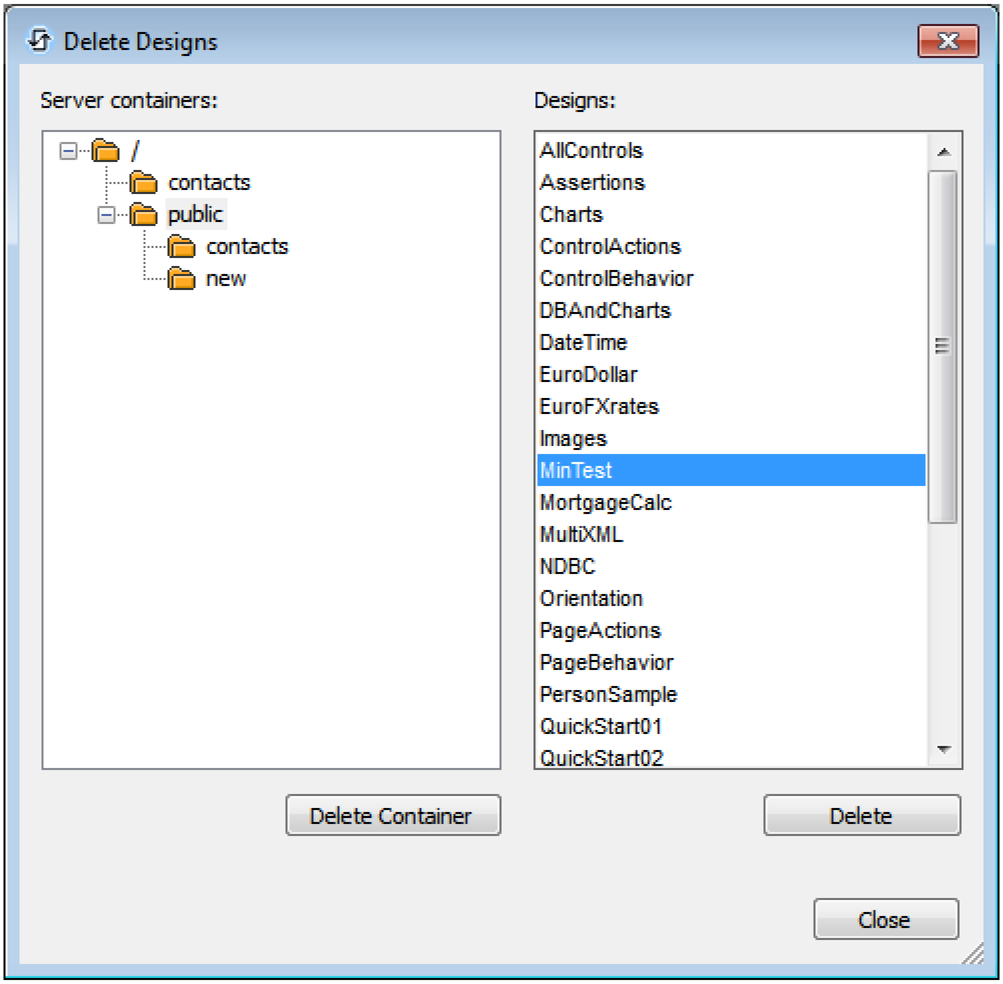
Domain: solutions.mt.altova.com

The following details are required for connecting to the server:

- *Server name and port:* The port refers to the MobileTogether Server administrator port and must match the *Administrator Port setting* of MobileTogether Server. If you use SSL, make sure that you use the secure port of MobileTogether Server.

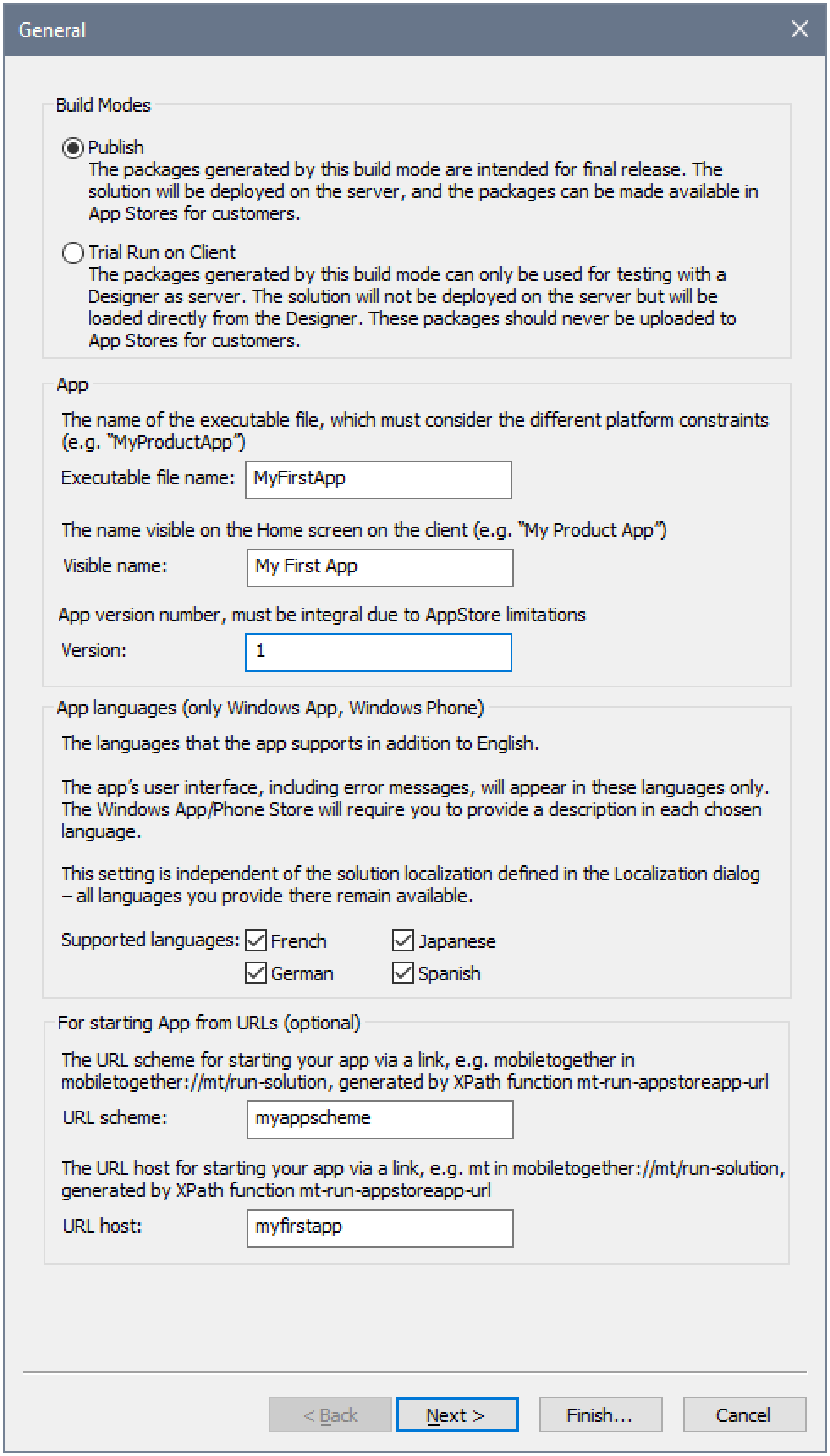
- *User name and password:* The user name and password of a user that has been given the privilege *Save workflow from designer*. MobileTogether Server users and their privileges are specified in the Users and Roles settings of MobileTogether Server.
- *Login:* Specify whether login is direct or as a domain user. (If login is as a domain user, then the domain-specific user name and password can be used.) From the options in the combo box, select *Directly* or the domain you wish to use. Only those domains are displayed in the combo box that have been configured on the server for active directory login. *For more information about configuring the server for active directory login, see the MobileTogether Server user manual.*

On clicking **Proceed**, a window showing MobileTogether Server folders (containers) and their solutions (designs) is displayed (*screenshot below*). Browse for the container or design you wish to delete, select it, and click **Delete Container** or **Delete**.



## 28.1.12 Generate Program Code for AppStore Apps

Opens the Generate Program Code Wizard[1472] for creating AppStore Apps[1471] (*screenshot below*). For a detailed description of how to use the wizard and how to generate appstore apps, see the section AppStore Apps[1471].



General

Build Modes

● Publish
The packages generated by this build mode are intended for final release. The solution will be deployed on the server, and the packages can be made available in App Stores for customers.

○ Trial Run on Client
The packages generated by this build mode can only be used for testing with a Designer as server. The solution will not be deployed on the server but will be loaded directly from the Designer. These packages should never be uploaded to App Stores for customers.

App

The name of the executable file, which must consider the different platform constraints (e.g. "MyProductApp")

Executable file name: MyFirstApp

The name visible on the Home screen on the client (e.g. "My Product App")

Visible name: My First App

App version number, must be integral due to AppStore limitations

Version: 1

App languages (only Windows App, Windows Phone)

The languages that the app supports in addition to English.

The app's user interface, including error messages, will appear in these languages only. The Windows App/Phone Store will require you to provide a description in each chosen language.

This setting is independent of the solution localization defined in the Localization dialog – all languages you provide there remain available.

Supported languages: ☑ French ☑ Japanese ☑ German ☑ Spanish

For starting App from URLs (optional)

The URL scheme for starting your app via a link, e.g. mobiletogether in mobiletogether://mt/run-solution, generated by XPath function mt-run-appstoreapp-url

URL scheme: myappscheme

The URL host for starting your app via a link, e.g. mt in mobiletogether://mt/run-solution, generated by XPath function mt-run-appstoreapp-url

URL host: myfirstapp

After you finish the wizard, the workflow will be deployed to the server and the program code for the selected app formats will be generated.

## 28.1.13 Send by Mail

*Icon*

*Description*

Sends the active MobileTogether Design document (MTD document) as an email attachment. On selecting the command, an email is opened with the active MTD document attached. Enter the name of the recipient/s in the *To:* field and subject information in the *Subject:* field of the email, then click the email application's **Send** command.

## 28.1.14 Print

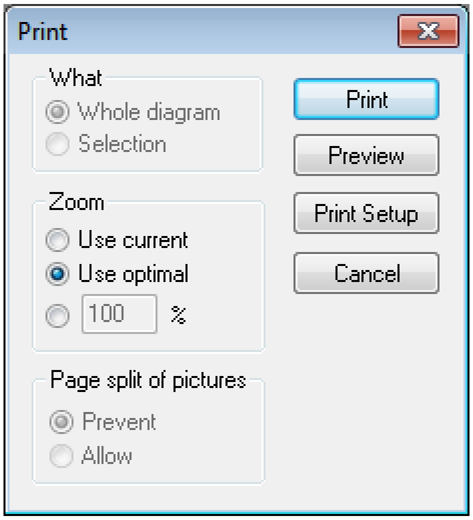
*Icon*

*Shortcut*

**Ctrl+P**

*Description*

Opens the Print dialog box (*screenshot below*), in which you can select printing options for printing the currently active document.



Print

What
● Whole diagram
○ Selection

Zoom
○ Use current
● Use optimal
○ 100 %

Page split of pictures
● Prevent
○ Allow

Options that are applicable to the current printing job are enabled. The following options are available:

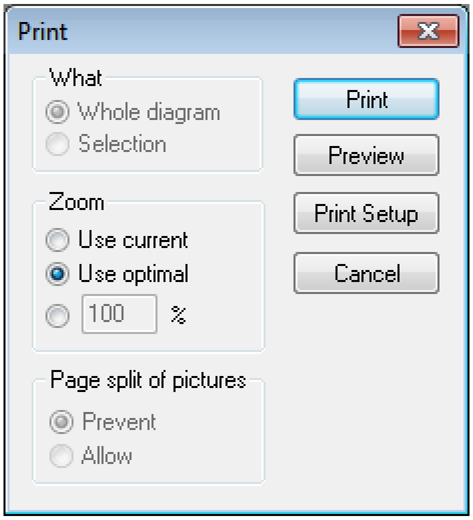
- *What:* Whether to print the entire design diagram or only the current selection.
- *Zoom:* The zoom level to use in the printout.
- *Page-split of pictures:* Whether images may be split (*Allow*) or not (*Prevent*).

## 28.1.15 Print Preview, Print Setup

▼ Print Preview

⊟ *Description*

Opens the print dialog box (*screenshot below*). Click **Preview** to see a preview of the currently active document according to the settings specified in the dialog 1582.



What
● Whole diagram
○ Selection

Zoom
○ Use current
● Use optimal
○ 100 %

Page split of pictures
● Prevent
○ Allow

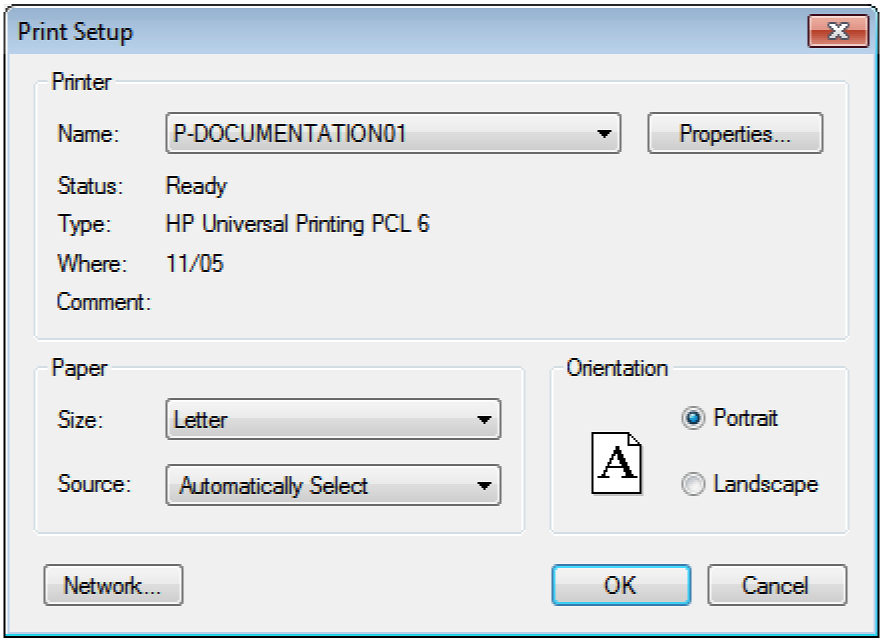
In Print Preview mode, the Print Preview toolbar at top left of the preview window provides print-related and preview-related options. The preview can be magnified or miniaturized using the **Zoom In** and **Zoom Out** buttons. When the page magnification is such that an entire page length fits in a preview window, then the **One Page / Two Page** button toggles the preview to one or two pages at a time. The **Next Page** and **Previous Page** buttons can be used to navigate among the pages. The toolbar also contains buttons to print all pages and to close the preview window.

**Note:** To enable background colors and images in Print Preview, do the following: (i) In the **Tools** menu of Internet Explorer, click **Internet Options**, and then click the Advanced tab; (ii) In the Settings box, under Printing, select the *Print background colors and images* check box, and (iii) then click **OK**.

▼ Print Setup

⊟ *Description*

Displays the printer-specific Print Setup dialog box, in which you specify such printer settings as paper format and page orientation. These settings are applied to all subsequent print jobs.



Name: P-DOCUMENTATION01

Status: Ready

Type: HP Universal Printing PCL 6

Where: 11/05

Comment:

Size: Letter

Source: Automatically Select

● Portrait

○ Landscape

## 28.1.16 Recent Files, Exit

Recent Files

*Description*

At the bottom of the **File** menu is a list of the nine most recently used files, with the most recently opened file shown at the top of the list. You can open any of these files by clicking its name. To open a file in the list using the keyboard, press **Alt+F** to open the **File** menu, and then press the number of the file you want to open.

Exit

*Description*

Quits MobileTogether Designer. If you have any open files with unsaved changes, you are prompted to save these changes. MobileTogether Designer also saves modifications to program settings and information about the most recently used files.

# 28.2 Edit

The **Edit** menu contains the following commands:

- Undo 1586
- Redo 1586
- Cut 1587
- Copy 1587
- Paste 1587
- Delete 1587
- Select All 1588

## 28.2.1 Undo, Redo

▼ Undo

⊟ *Icon*

↶

⊟ *Shortcut*

**Ctrl+Z**

⊟ *Description*

Supports unlimited levels of Undo. Every action can be undone and it is possible to undo one command after another. The Undo history is retained after using the **Save** command, enabling you go back to the state the document was in before you saved your changes. You can step backwards and forwards through this history using the **Undo** and **Redo** commands (*see **Redo** command below*).

▼ Redo

⊟ *Icon*

↷

⊟ *Shortcut*

**Ctrl+Y**

⊟ *Description*

Allows you to redo previously undone commands, thereby giving you a complete history of work completed. You can step backwards and forwards through this history using the **Undo** and **Redo** commands.

## 28.2.2 Cut, Copy, Paste, Delete

▼ Cut

⊟ *Icon*

⊟ *Shortcut*

**Ctrl+X** or **Shift+Del**

⊟ *Description*

Copies the selected text or items to the clipboard, deleting the selection from its current location.

▼ Copy

⊟ *Icon*

⊟ *Shortcut*

**Ctrl+C**

⊟ *Description*

Copies the selected text or items to the clipboard, *without* deleting the selection from its current location. The command can be used to duplicate data within MobileTogether Designer, or to move data to another application.

▼ Paste

⊟ *Icon*

⊟ *Shortcut*

**Ctrl+V**

⊟ *Description*

Inserts the contents of the clipboard at the current cursor position.

▼ Delete

⊟ *Icon*

☒

⊟ *Shortcut*

**Del**

⊟ *Description*

Deletes the currently selected text or items without placing them in the clipboard.

## 28.2.3 Select All

⊟ *Shortcut*

**Ctrl+A**

⊟ *Description*

Selects the contents of the entire document.

# 28.3 Project

The **Project** menu contains commands that apply to the entire project. It contains the following commands:

- Validate 1589
- Reload Page Source Structures 1589
- Global Variables 1590
- XPath/XQuery Functions 1591
- Action Groups 1594
- Style Sheets 1597
- Rich Text Style Sheets 1598
- Cache Overview 1598
- Localization 1600
- Simulation Language 1606
- Maintain OAuth Settings 1606
- Import OAuth Settings 1607
- iOS Push Notification Button Sets 1608
- In-App Purchase Products 1609

## 28.3.1 Validate

⊟ *Icon*

⊟ *Description*

Validates the currently active project. If a project contains multiple pages, all the pages are validated. Validation results are reported in the Messages Pane 278 in terms of the number of errors and warnings. If an error or warning is detected, a message about each is displayed.

If there is a validation error because of a missing page source on a page, then the Messages pane will offer a quick-fix to add the missing page source to that page. On clicking the quick-fix, a dialog appears that enables you to add the missing page source to not only the selected page but to all other pages where this quick-fix is applicable. This enables you to fix missing-page-source-errors faster.

## 28.3.2 Reload Page Source Structures

Reloads the structures of all page sources of the project. This command is useful if you want to reload all pages sources at once. Additional options for reloading page source structures are available in the File tab of the Options dialog 1666.

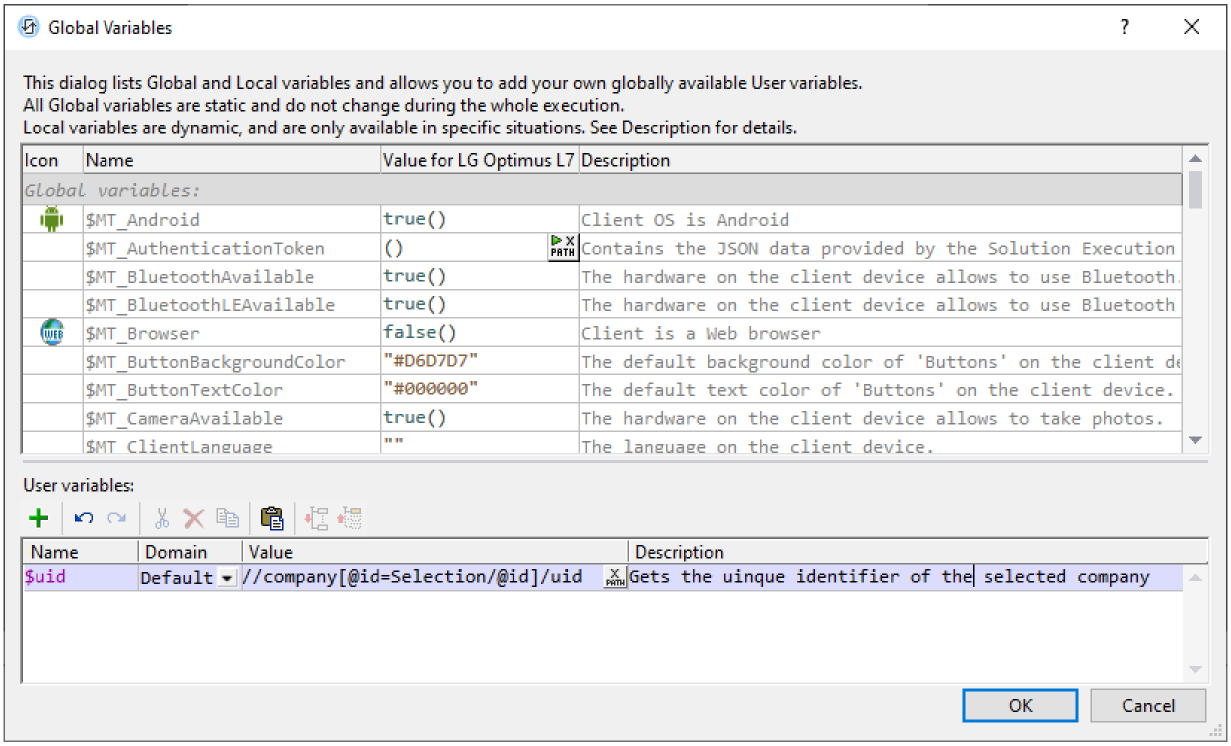
## 28.3.3 Global Variables

*Icon*

*Description*

Opens the Global Variables dialog (*screenshot below*). Global variables are available to the designer in programming contexts, such as XPath or XQuery, everywhere in the project. MobileTogether Designer provides a standard library of global variables, which are listed in the upper pane of the dialog. The values of the global variables are assigned by MobileTogether during simulation and when the app runs on the client device. Global variables are of three types, and the list of variables in the dialog is divided into three parts for these three types:

- Static-value variables (called *Global Variables* in the dialog): These variables contain information about the mobile device. Values of these variables do not change during the execution of the project. Notice that the *Value* column in the upper pane specifies the currently selected mobile device. The listed values are for that particular device. For simulations, the client device is considered to be the device selected in the Device Selector combo box[254]. For example, the variable `$MT_Android` has a value of **`true`** when the mobile device being used is an Android. (Device information is sent by the device as part of standard mobile communication procedures.)
- Dynamic-value variables (called *Local Variables* in the dialog): These variables contain information related to the design page and its controls. Their values could change during execution. For example, the `$MT_ControlNode` variable has different values according to which node is the page source link of the current control at a given time during project execution.
- *User variables*: In addition to the standard library of global variables, you can add your own global variables (called *User Variables* in the dialog) in the lower pane of the dialog. You can give a user variable any value, and this value can then be used subsequently in any XPath/XQuery expression in the project.



Global Variables

This dialog lists Global and Local variables and allows you to add your own globally available User variables.
All Global variables are static and do not change during the whole execution.
Local variables are dynamic, and are only available in specific situations. See Description for details.

| Icon | Name | Value for LG Optimus L7 | Description |
|---|---|---|---|
| *Global variables:* | | | |
| | $MT_Android | true() | Client OS is Android |
| | $MT_AuthenticationToken | () | Contains the JSON data provided by the Solution Execution |
| | $MT_BluetoothAvailable | true() | The hardware on the client device allows to use Bluetooth. |
| | $MT_BluetoothLEAvailable | true() | The hardware on the client device allows to use Bluetooth |
| WEB | $MT_Browser | false() | Client is a Web browser |
| | $MT_ButtonBackgroundColor | "#D6D7D7" | The default background color of 'Buttons' on the client de |
| | $MT_ButtonTextColor | "#000000" | The default text color of 'Buttons' on the client device. |
| | $MT_CameraAvailable | true() | The hardware on the client device allows to take photos. |
| | $MT_ClientLanguage | "" | The language on the client device. |

User variables:

| Name | Domain | Value | Description |
|---|---|---|---|
| $uid | Default | //company[@id=Selection/@id]/uid | Gets the uinque identifier of the selected company |

To add a user variable, in the lower pane, do the following:

1. Click the **Append** or **Insert** icons (located in the pane's toolbar) to add a line to the list.
2. Enter the name of your new variable (in the *Name* column) and give the variable a description (*Description* column). *See screenshot above*.
3. Click in the *Value* field to bring up the Edit XPath/XQuery Expression dialog, and enter the XPath expression that determines the value of this variable.
4. Select an icon to help identify the new variable as belonging to a particular group.
5. Click **OK** to finish. The variable is added as a global variable, and can be used in programming contexts.

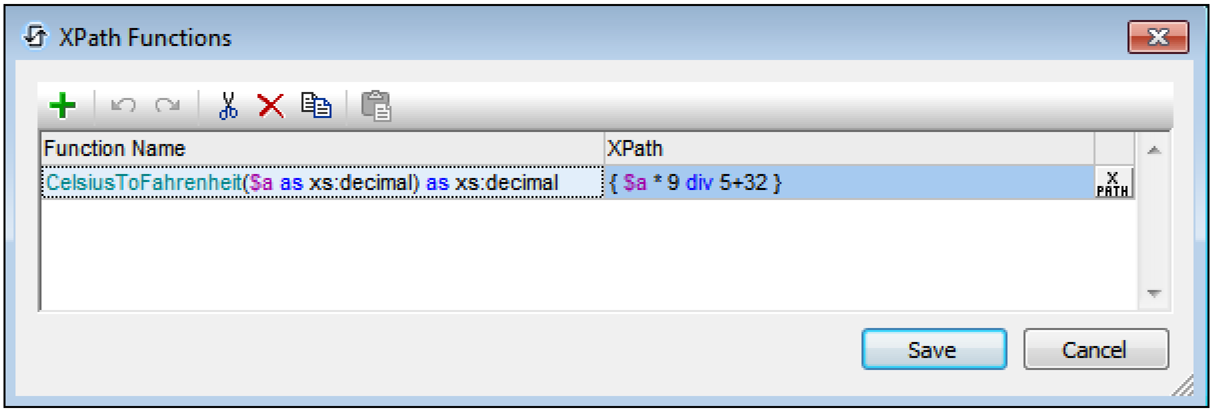
See the section, Global Variables 1298 for a description of predefined global variables.

## 28.3.4 XPath/XQuery Functions

⊟ *Icon*

⊟ *Description*

Opens the XPath Functions dialog, which lists all the user-defined XPath functions in the project. These XPath functions can be used in all XPath expressions in the project. You can add and delete functions using the corresponding icons in the toolbar of the dialog. To edit the definition of a function, click the function's **Edit XPath Expression** button.

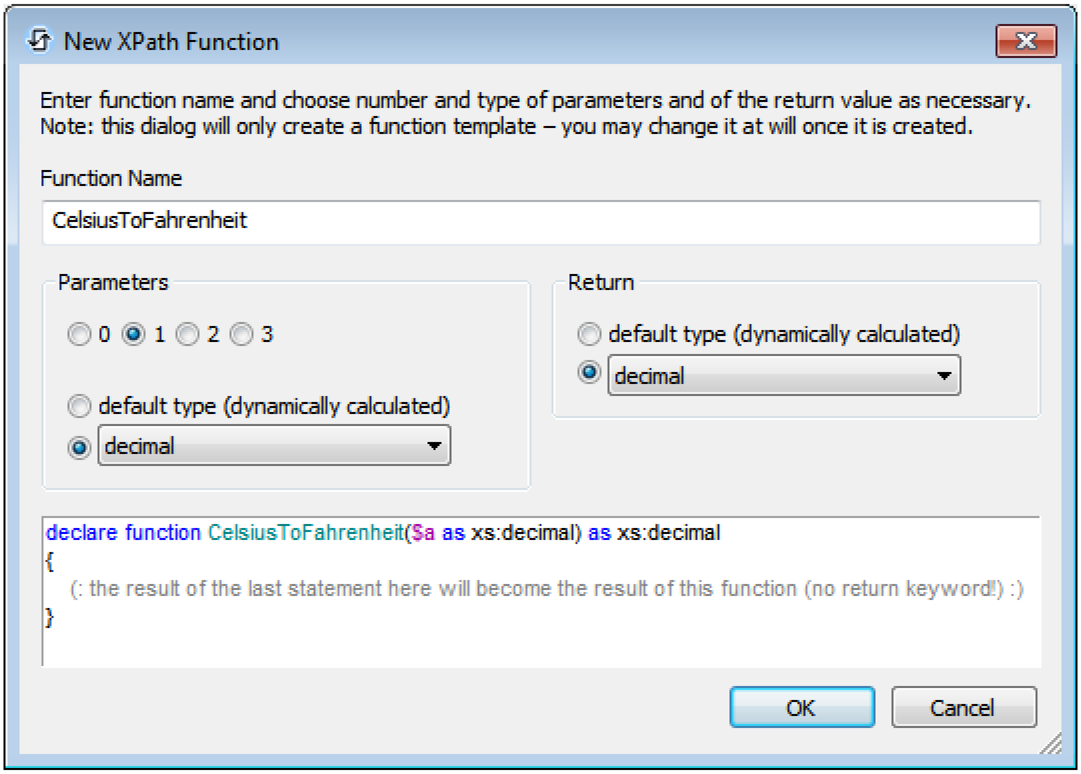


| Function Name | XPath |
|---|---|
| CelsiusToFahrenheit($a as xs:decimal) as xs:decimal | { $a * 9 div 5+32 } |

## Add a new user-defined XPath function

Adding a new user-defined function involves two steps: (i) declaring the function, and (ii) defining the function.

To add a new function, do the following, click **Add** in the toolbar of the dialog (*see screenshot above*). This displays the New XPath Function dialog (*screenshot below*).



New XPath Function

Enter function name and choose number and type of parameters and of the return value as necessary.
Note: this dialog will only create a function template – you may change it at will once it is created.

Function Name: CelsiusToFahrenheit

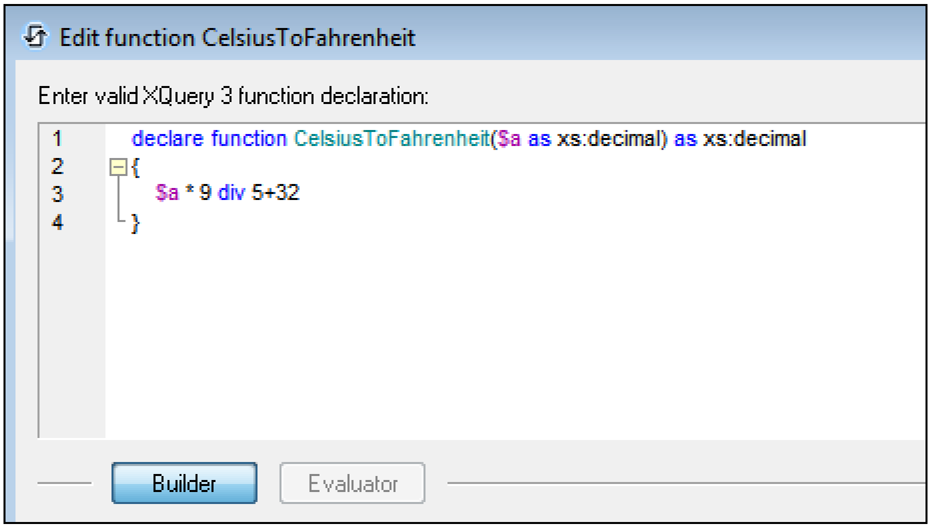
Parameters: ○ 0 ● 1 ○ 2 ○ 3

○ default type (dynamically calculated)
● decimal

Return:
○ default type (dynamically calculated)
● decimal

```
declare function CelsiusToFahrenheit($a as xs:decimal) as xs:decimal
{
   (: the result of the last statement here will become the result of this function (no return keyword!) :)
}
```

In this dialog, you can declare the name of the function, specify the number of function parameters (arguments) and their types, and specify the return type of the function. In the screenshot above we have declared a function to convert a decimal number from Celsius to Fahrenheit. The function takes one parameter, which is the input Celsius value as a decimal. It will output a decimal value, the Fahrenheit temperature. What the function does is defined in the next step. After declaring the function (*screenshot above*), click **OK**. This displays the Edit Function dialog (*screenshot below*), which contains the template of the newly declared function and in which you can now define the function.


Edit function CelsiusToFahrenheit
Enter valid XQuery 3 function declaration:
Builder
Evaluator

```
declare function CelsiusToFahrenheit($a as xs:decimal) as xs:decimal
{
   $a * 9 div 5+32
}
```

Enter the definition of the function within the braces. In the definition shown in the screenshot above, **$a** is the input parameter. Click **OK**. when done. The function will be added to the list of user-defined functions in the XPath Functions dialog and can be used in all XPath expressions in the project.

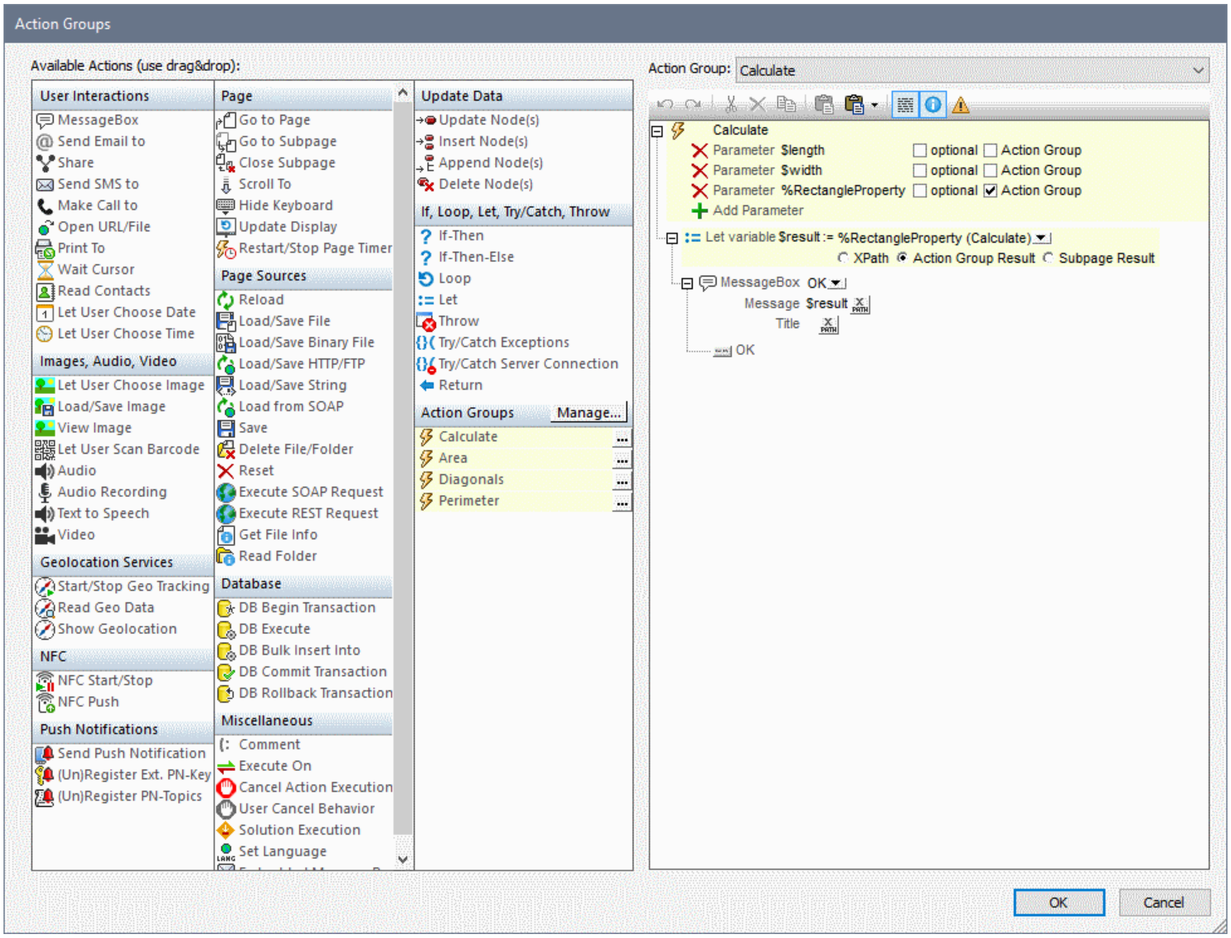
**Note:** User-defined XPath functions do not need to be placed in a separate namespace. Consequently, no namespace prefix is needed when defining or calling a user-defined function. The XPath default namespace 310 is used for all XPath/XQuery functions, including extension functions and user-defined functions 1293. In order to avoid ambiguities involving built-in functions, we recommend that you capitalize user-defined functions.

## 28.3.5 Action Groups

⊟ *Icon*

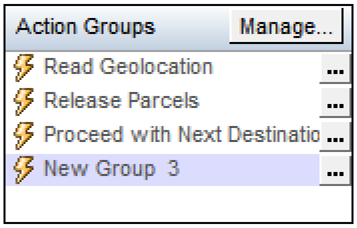
⊟ *Description*

Displays the Action Groups dialog (*screenshot below*), in which action groups can be created and edited.



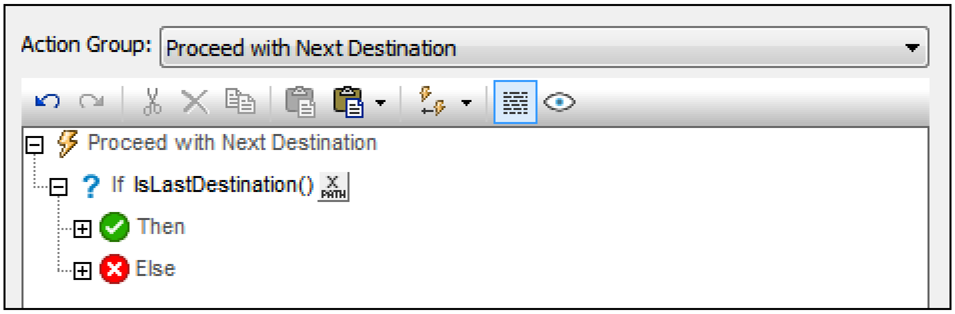
To create an Action Group, do the following:

1. Click **Manage** in the Action Groups pane (*see screenshot below*).



2. In the Manage Group Actions dialog that appears, click **Add** in the toolbar of the dialog. An Action Group with a default name is added to the list of Action Groups in this dialog.
3. Double-click the name of Action Group to a suitable name to change it, and click **OK**. The Manage Action Groups dialog closes, and the new Action Group is added to the list of groups in the Action Groups pane (*see screenshot above*).

4. In the Action Groups pane (*screenshot above*), click the **Edit** icon of the Action Group you want to edit. The group's contents are displayed in the pane on the right (*see screenshot below*). The details of an Action Group can also be displayed by selecting the group in the Action Group combo box (*see screenshot below*).



Action Group: Proceed with Next Destination

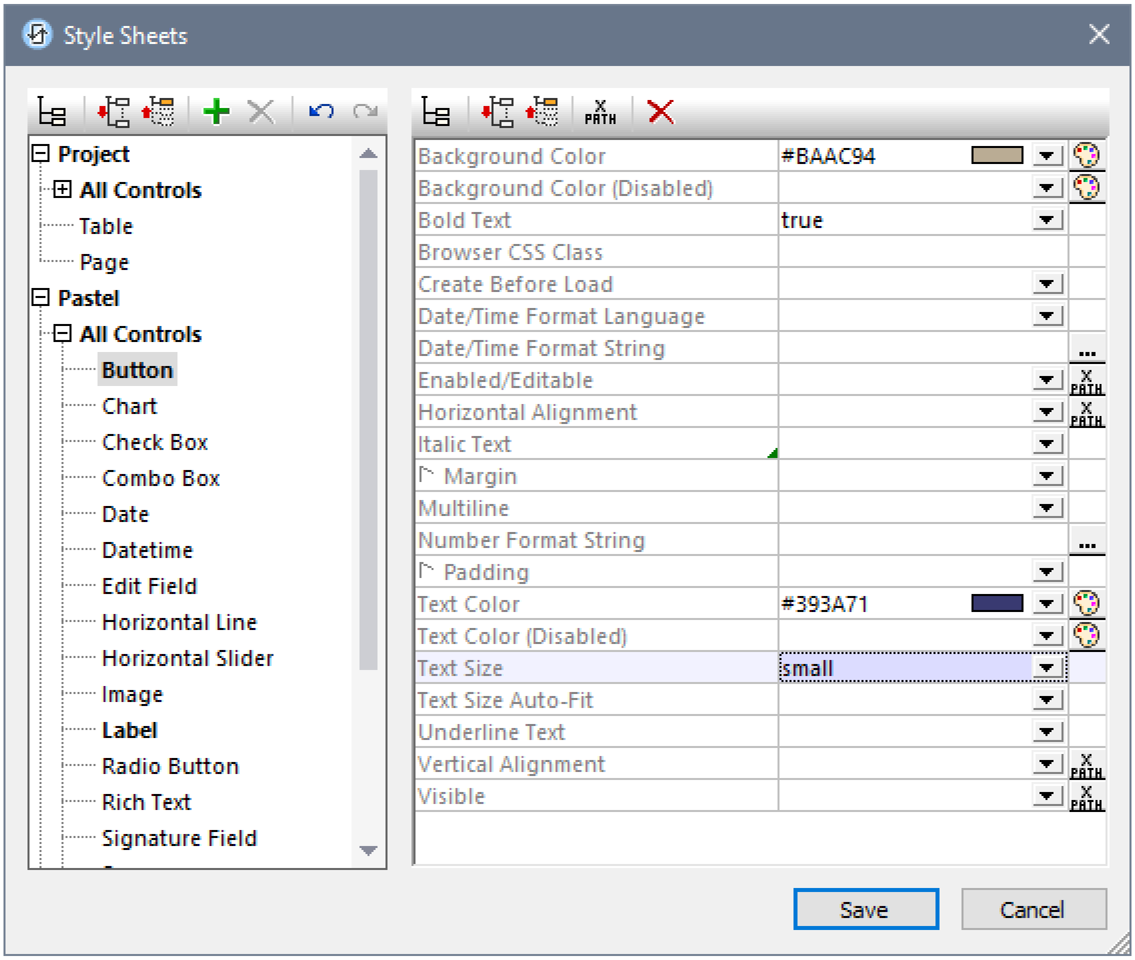
5. To edit the contents of the active Action Group in the right-hand pane, drag and drop actions and and action groups from the Available Actions pane on the left.
6. Click **OK** to finish. The Action Group you have edited is available for use.

**Note:** The last selection in this dialog is remembered. As a result, the dialog is always reopened with the last selection highlighted.

**Note:** An action group can be edited at any time. Click its **Edit** button to display it in the right-hand editor pane; alternatively, select it in the combo box above the editor pane.

## 28.3.6 Style Sheets

Displays the Style Sheets dialog (*screenshot below*), in which you can manage style sheets and define the styles in the different style sheets.



- Project
  - All Controls
  - Table
  - Page
- Pastel
  - All Controls
    - Button
    - Chart
    - Check Box
    - Combo Box
    - Date
    - Datetime
    - Edit Field
    - Horizontal Line
    - Horizontal Slider
    - Image
    - Label
    - Radio Button
    - Rich Text
    - Signature Field

| | |
|---|---|
| Background Color | #BAAC94 |
| Background Color (Disabled) | |
| Bold Text | true |
| Browser CSS Class | |
| Create Before Load | |
| Date/Time Format Language | |
| Date/Time Format String | |
| Enabled/Editable | |
| Horizontal Alignment | |
| Italic Text | |
| Margin | |
| Multiline | |
| Number Format String | |
| Padding | |
| Text Color | #393A71 |
| Text Color (Disabled) | |
| Text Size | small |
| Text Size Auto-Fit | |
| Underline Text | |
| Vertical Alignment | |
| Visible | |

You can carry out the following actions to manage your library of style sheets:

- *Add a user-created style sheet:* Click **Add Style Sheet** ✚.
- *Rename a user-created style sheet:* Double-click the style sheet name and edit it.
- *Delete a user created style sheet:* Click **Delete Style Sheet** ☒.

**Note:** The project style sheet, named *Project* (*see screenshot above*), is available by default. You cannot rename or delete it. See the section Style Sheet Type and Scope 1320 for more information about the *Project* style sheet and user-created style sheets.

If, in the left pane, you select a control, table, or page, the styles available for that component are defined in the right-hand pane. To set a style value, enter the desired value, or select a value from the style's combo box options, or enter an XPath expression that evaluates to a style value. *See* Style Sheet Properties 1328 *for more information*.

Click **Save** when you are  done.

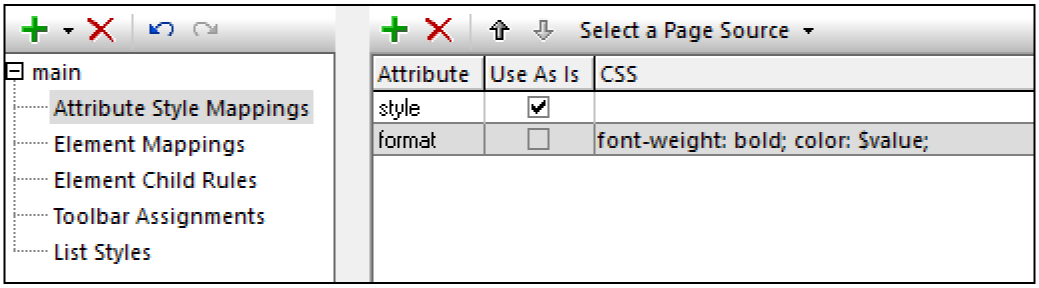
**Note:** Each pane has an icon that enables/disables the display of non-empty items (that is, those items for which a value (or at least one value) has been defined). Displaying only the non-empty items is useful when you wish to see a list of only the styles that have been defined; for example, when you want an overview of currently defined styles. The left-hand pane also has toolbar icons for (i) expanding all items, and (ii) collapsing all items.

For an overview of how style sheets work, see the section Style Sheets 1318.

## 28.3.7 Rich Text Style Sheets

Opens the Rich Text Style Sheets dialog (*screenshot below*), in which you can define:

- Styling rules for text in the solution display
- Mappings of styles from the page source to the solution display, and vice versa



- main
  - Attribute Style Mappings
  - Element Mappings
  - Element Child Rules
  - Toolbar Assignments
  - List Styles

| Attribute | Use As Is | CSS |
|---|---|---|
| style | ☑ | |
| format | ☐ | font-weight: bold; color: $value; |

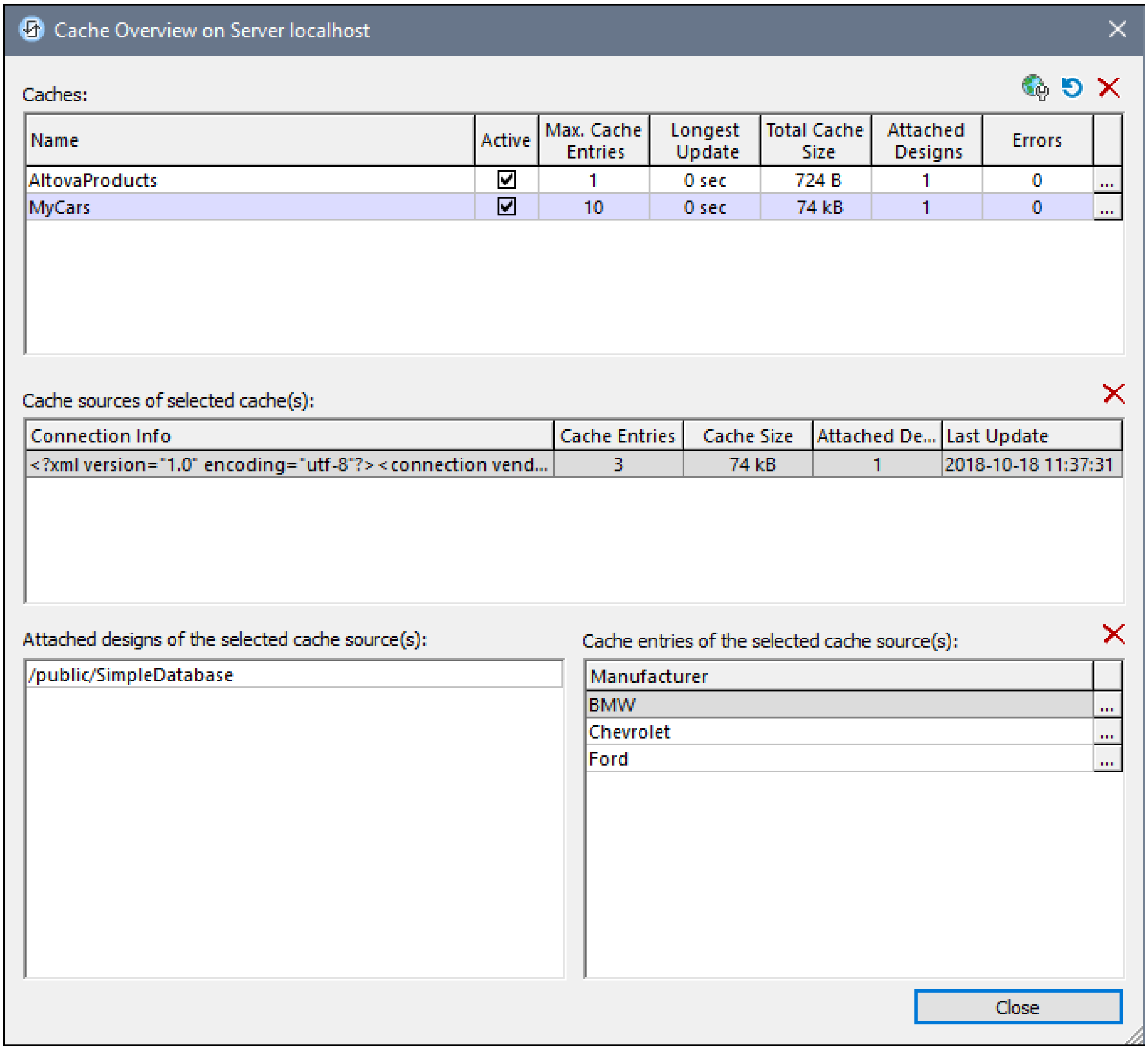
For a description of the Rich Text Style Sheets dialog, see the Rich Text 1228 section.

## 28.3.8 Cache Overview

⊟ *Icon*

⊟ *Description*

Connects to MobileTogether Server and provides an overview of server caches in the Cache Overview dialog (*screenshot below*).



Cache Overview on Server localhost

Caches:

| Name | Active | Max. Cache Entries | Longest Update | Total Cache Size | Attached Designs | Errors | |
|---|---|---|---|---|---|---|---|
| AltovaProducts | ☑ | 1 | 0 sec | 724 B | 1 | 0 | ... |
| MyCars | ☑ | 10 | 0 sec | 74 kB | 1 | 0 | ... |

Cache sources of selected cache(s):

| Connection Info | Cache Entries | Cache Size | Attached De... | Last Update |
|---|---|---|---|---|
| <?xml version="1.0" encoding="utf-8"?><connection vend... | 3 | 74 kB | 1 | 2018-10-18 11:37:31 |

Attached designs of the selected cache source(s):

/public/SimpleDatabase

Cache entries of the selected cache source(s):

| Manufacturer | |
|---|---|
| BMW | ... |
| Chevrolet | ... |
| Ford | ... |

The dialog provides, in the top pane, an overview of all the caches on the server. In the dialog, you can do the following:

- Activate/deactivate a cache (in the top pane).
- Delete a cache (in the top pane) by selecting it and clicking **Delete**.
- Select a cache in the top pane to display its details in the other (subsidiary) panes of the dialog.
- Delete a cache item in a subsidiary pane by selecting it and clicking the subsidiary pane's **Delete** button.
- Reconfigure a cache by clicking the **Additional Dialog** button (at the right-hand side of the cache entry in the top pane), and editing the cache's configuration.
- View a log of cache entry updates by clicking the **Additional Dialog** button in the Cache Entries pane (located at the bottom right of the dialog). In the screenshot above the `MyCars` cache has

three cache entries, which are listed in the Cache Entries pane.

See the section Caching[376] for more information about caching.

## 28.3.9 Localization

⊟ *General description*

Displays the Localization dialog (*screenshot below*), in which you can localize (translate) strings. These would be strings that either appear in various **controls** (for example, the text of a button) or are related to **controls** in other ways (for example, the dropdown list values of combo boxes). In addition to strings related to controls, any **custom string** can be localized and subsequently inserted at any location in the design via the `mt-load-string`[1262] function.

- **Control-related strings** are displayed in the upper pane (*see screenshot below*). The columns of this pane are, respectively, from left to right: (i) the page in which the control appears; (ii) the control's name; (iii) the control property that contains the text string; (iv) the text string in the default language; additional columns contain the text string in the localized languages, one column per language.
- **Custom strings** are displayed in the lower pane. The columns of this pane are, respectively, from left to right: (i) the custom string's name; (ii) the custom string in the default language; additional columns contain the custom string in the localized languages, one column per language.

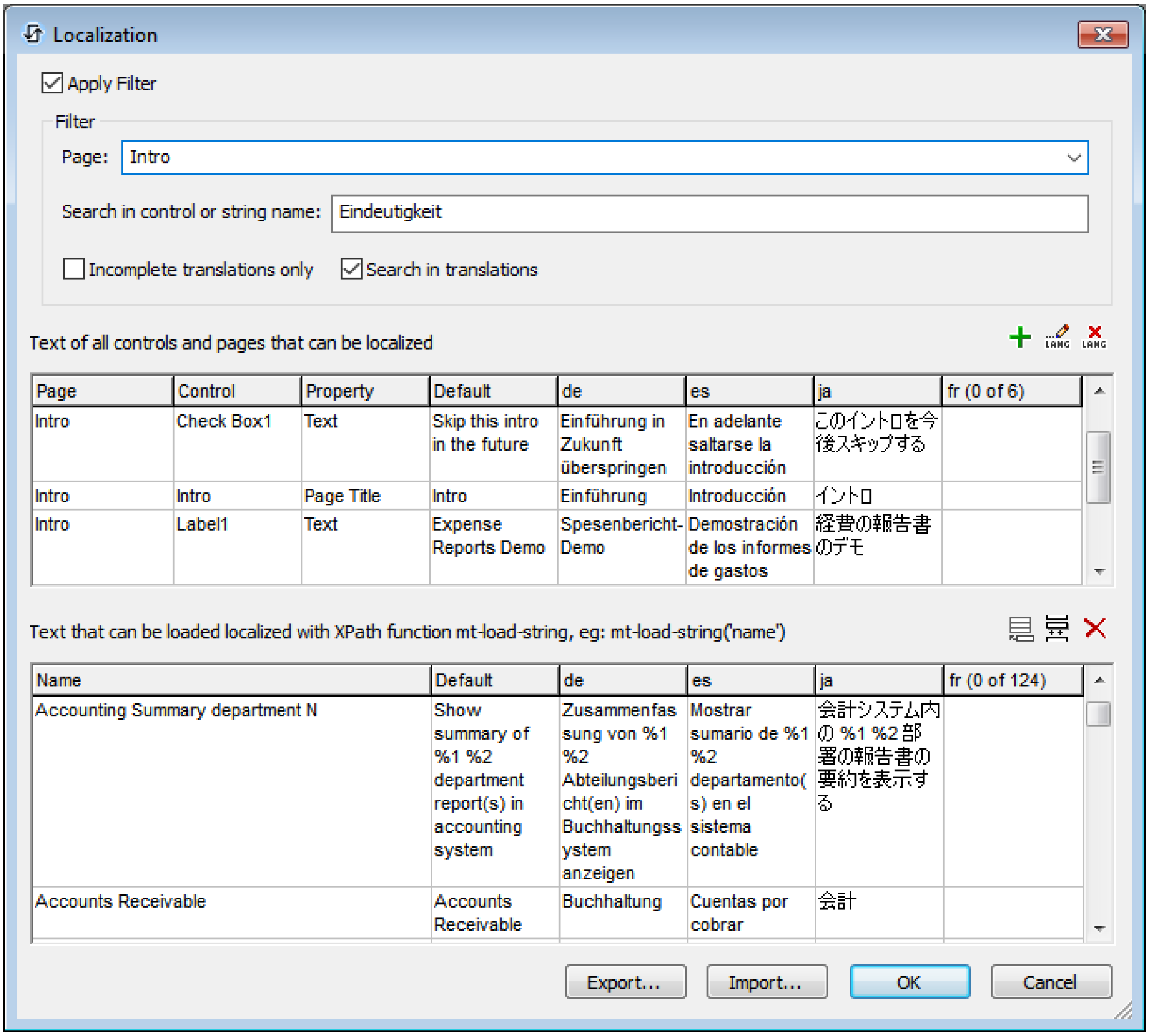
When the dialog opens, all strings are listed by control, and the first string of the control that had the focus (when the dialog was opened) will be selected in the dialog. To edit, double-click in a field or press **F2**. You can navigate to the next field by using the **Tab** key.

You can add a column for a localization language with the **Add Language** icon. In the dialog that appears, enter the language code (used for internal) and language name (used in MobileTogether user interfaces). You can add as many columns as you like. To localize a control-related string or custom string in a particular language, enter the translation in the column for that language. To save translations, click **OK**.

All strings (control-related and custom) that have been localized in the Localization dialog will be used in the localized versions of the solution. If the language setting on the mobile device specifies a `language-country` variant (for example, **`es-US`** or **`fr-CH`**), then the solution's language is selected according to the cascading order given below:

1. If the solution contains a matching `language-country` (**`es-US`** or **`fr-CH`**) localization, then strings of this localization are used where these exist
2. If no matching `language-country` localization (**`es-US`** or **`fr-CH`**) exists for a string, then the localized `language` (**`es`** or **`fr`**) string is used—if one exists
3. If no `language-country` localization (**`es-US`** or **`fr-CH`**) or `language` localization (**`es`** or **`fr`**) exists for a string, then the default language of the solution is used for that string

If you wish to see a simulation in any of the languages for which localized strings are defined, set the simulation language via the **Project | Simulation Language**[1606] command, and then run a simulation[1355]
.



Localization

☑ Apply Filter

Filter

Page: Intro

Search in control or string name: Eindeutigkeit

☐ Incomplete translations only ☑ Search in translations

Text of all controls and pages that can be localized

| Page | Control | Property | Default | de | es | ja | fr (0 of 6) |
|---|---|---|---|---|---|---|---|
| Intro | Check Box1 | Text | Skip this intro in the future | Einführung in Zukunft überspringen | En adelante saltarse la introducción | このイントロを今後スキップする | |
| Intro | Intro | Page Title | Intro | Einführung | Introducción | イントロ | |
| Intro | Label1 | Text | Expense Reports Demo | Spesenbericht-Demo | Demostración de los informes de gastos | 経費の報告書のデモ | |

Text that can be loaded localized with XPath function mt-load-string, eg: mt-load-string('name')

| Name | Default | de | es | ja | fr (0 of 124) |
|---|---|---|---|---|---|
| Accounting Summary department N | Show summary of %1 %2 department report(s) in accounting system | Zusammenfassung von %1 %2 Abteilungsbericht(en) im Buchhaltungssystem anzeigen | Mostrar sumario de %1 %2 departamento(s) en el sistema contable | 会計システム内の %1 %2 部署の報告書の要約を表示する | |
| Accounts Receivable | Accounts Receivable | Buchhaltung | Cuentas por cobrar | 会計 | |

## Apply Filter

The following filters are available in the Localization dialog; they can be combined:

- *Page:* Select a page from the dropdown list to see the strings of only that page. To select all pages, leave the option blank or select the empty entry. This filter applies to control-related strings (upper pane) only.
- *Control-related strings **and** custom strings:* Enter the text to search for. All control names (second column of upper pane) **and** custom string names (first column of lower pane) that contain the search text are displayed. This filter applies to both panes.
- *Incomplete translations only:* Check this option to show only incomplete translations. An incomplete translation is a string with at least one localization missing. This filter applies to both panes. Note that, if a translation is entered while this option is selected, then the

display must be refreshed in order to update the filtering. You can refresh the display by deselecting and then re-selecting the option.

- *Search in translations:* If this check box is selected, the search (for the term entered) is carried out in both the default language and translations. The search is carried out while typing the search term.

## ⊟ *Adding, renaming, and deleting language columns*

These icons are located above the upper pane, on the right-hand side.

| | | |
|---|---|---|
| + | **Add Language** | Displays the Add Language dialog, in which you can (i) enter or select a language's ISO code (`ISO639-ISO3166`), and (ii) enter a language name. On clicking **OK**, a column with that language's code is added to both (upper and lower) panes. (*See also the related* **`mt-available-languages`**[1262] *XPath function.*) |
| LANG | **Rename Language** | Place the cursor (in a row in either pane) in the language column you want to rename, then click **Rename Language**. The Change Language dialog is displayed with the language to change being pre-selected in the edit field. Change the language name and click **OK**. The column name will be changed in both panes. You can also modify the name of the default language; but note that the default language has no language code. |
| ✕ LANG | **Delete Language** | Place the cursor (in a row in either pane) in the language column you want to delete, then click **Delete Language**. On being prompted whether you wish to delete the column, click **Yes** to delete the column and its entries, **No** to cancel. |

## ⊟ *Custom strings (lower pane)*

Custom strings can be any text you wish to use in the design. They are managed in the lower pane by using the icons located above the pane, on the right-hand side.

| | | |
|---|---|---|
| ▤ | **Add Localization String** | Adds a row for a custom string. The newly added custom string is created with a default name, which you can edit. Click in the name to start editing. The name is used to reference the custom string from XPath expressions. *See below.* |
| ▤ | **Duplicate Row** | Duplicates the custom string that is currently selected in the lower pane (name and all localizations). This saves time if you wish to add a string that is almost the same as an already existing string. When names of two strings are the same, both names are displayed in red. Edit the duplicate string as needed. |
| ✕ | **Remove Localization String** | Removes the currently selected custom string. |

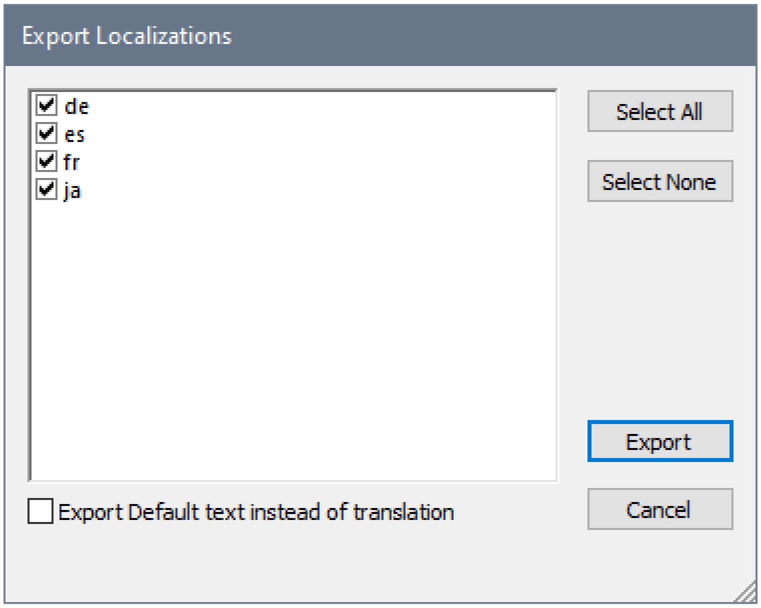
To use a custom string in the design, use the `mt-load-string`[1262] function in an XPath expression, like this: `mt-load-string('NameOfString')`.

If you change the name of a string, then a dialog appears asking whether you want to rename the string in all the `mt-load-string()`calls to it. Note that this will only apply if the string is the first parameter of the function and it is not passed to the function via a variable.

⊟ *Exporting and importing localizations*

Localized strings can be exported to XML files. You can choose to export one or more languages to a single file (*see screenshot below*). The advantage is that translators can work independently with their separate XML language files. The translated strings in the XML files can then be imported into the project, and will be placed in their corresponding localization-language columns in the Localization dialog.

When you click **Export** in the Localization dialog (*see screenshot at the beginning of this topic*), the Export Localizations dialog (*screenshot below*) appears. The languages that are displayed in the dialog are the languages that have been defined as the localization languages of the project. Select one or more languages to export.



If the *Export default text instead of translation* option is selected, then the exported file will contain the default-language text—instead of already-completed translations—as the values of all the localization-language attributes (*see XML file listings below*). The project, however, will retain the translations. Exporting default-language text can be useful if you use translation tools that work with translation memories. This is because, when an XML file is imported into the translation tool, the translation memory will automatically translate all the imported strings on the basis of what's in its memory. So, if any of the newly imported default text strings were previously translated, then they will simply be re-translated from the translation memory. Additionally, however, if previously untranslated strings contain words or phrases that are in the translation memory, then these words or phrases will be automatically translated. Any remaining words can then be translated manually. When the translated XML file is imported into the project, the translations are entered into the corresponding localization-language columns of the Localization dialog.

Clicking **Export** in the Export Localizations dialog, displays a Save dialog in which you can specify the name and location of the XML file.

*XML listing of one-language export*

```
<Localizations version="1">
  <Controls>
    <Control default="Select Employee" property="Page Title" name="Select
Employee" id="2">
      <Languages de="Mitarbeiter auswählen"/>
    </Control>
    ...
  </Controls>
  <Strings>
    <String default="Admin" name="Role-A Name">
      <Languages de="Admin"/>
    </String>
    ...
  </Strings>
</Localizations>
```

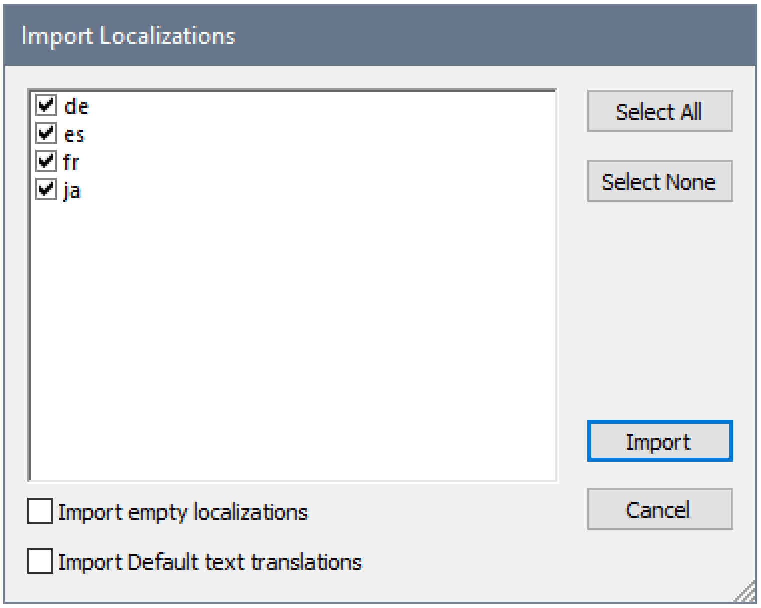
*XML listing of multiple-language export*

```
<Localizations version="1">
  <Controls>
    <Control default="Select Employee" property="Page Title" name="Select
Employee" id="2">
      <Languages de="Mitarbeiter auswählen" es="Seleccionar empleado" fr=""/>
    </Control>
    ...
  </Controls>
  <Strings>
    <String default="Admin" name="Role-A Name">
      <Languages de="Admin" es="Administración" fr=""/>
    </String>
    ...
  </Strings>
</Localizations>
```

**Note:** Strings that have not been translated into a localization language are exported as empty strings (if the export of default-language text (instead of translations) has not been enabled).

*Importing a translated XML file*

To import a translated XML file, click **Import** in the Localization dialog (*see screenshot at the beginning of this topic*). This displays an Open dialog in which you can select the XML file to import. When you click **Open**, the Import Localizations dialog (*screenshot below*) is displayed. The languages displayed are those languages found in the XML file that have corresponding columns in the Localization dialog. If a localization language is present in the XML file, but no corresponding localization language name is found in the project, then that language is not displayed.



☑ de
☑ es
☑ fr
☑ ja

□ Import empty localizations
□ Import Default text translations

- Select the language/s you wish to import.
- If you select the *Import empty localizations* option, then empty localization strings in the XML file will overwrite all existing localization strings, even if the existing strings are non-empty.
- If you select the *Import default text translations* option, then incoming translations that are the same as the default-language text will be imported (as with the `Admin` string for German (`de`) in the listing above). Otherwise, nothing will be imported for such fields, and the existing text in these fields will not be modified. (The *Import default text translations* option is enabled only if the option to export the strings of the default language was selected during the Export process; *see description of Export above*.)
- Click **Import** to finish.

The translated language strings are imported into the project and entered into the corresponding localization-language column/s of the Localization dialog. Note that the structure and content of the imported XML file must be such that MobileTogether Designer can correctly process the XML file. It is therefore important not to change the values of any other attributes besides the localization-language attributes.

**Note:** In the XML file, if a translated string is identical to the default-language string, then the translated string is **not** imported and the corresponding entry in the localization-language column is empty. When the solution is run, since there is no localization-language entry, the default-language string will be used. In this way, duplications of strings across languages are avoided.

## 28.3.10 Simulation Language

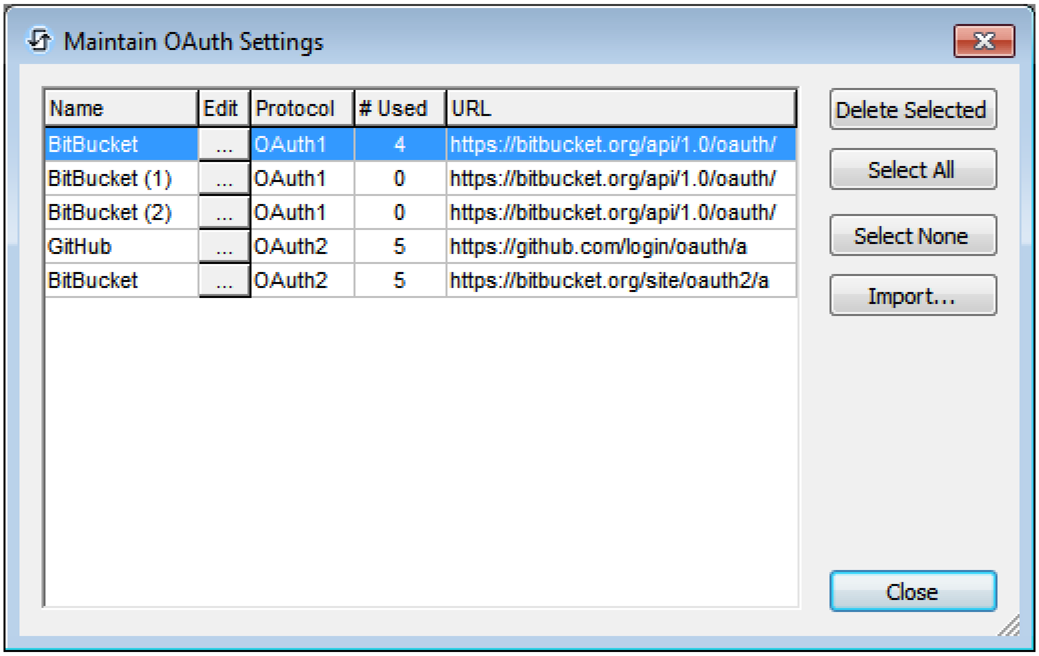
When a project is localized[1600], its text strings are translated into the target language. As a result, the localized project will be available to the end user as a solution in the target language.

If a project has been localized[1600] into one or more languages, these languages are available in the submenu of the **Simulation Language** command. In this submenu, you can select the language used for project simulations. The localized strings of the selected language will be used for all simulations of the project till the simulation language is changed.

## 28.3.11 Maintain OAuth Settings

REST requests made in MobileTogether Designer can be authenticated with OAuth. See the section REST Request Settings[332] for a description of how to do this.

You can create multiple OAuth setting definitions in a MobileTogether Designer project *(see the documentation here*[332]). These are stored in a pool, and you can use a definition from the pool for authenticating REST requests defined anywhere in the document. The Maintain OAuth Settings dialog (*screenshot below*) lets you manage the OAuth definitions of the active project. The dialog displays all the OAuth settings definitions that are currently in the active project's pool of definitions. You can delete definitions from the pool, import definitions from other open MobileTogether Designer projects, copy definitions to the clipboard, and paste definitions from the clipboard.



| Name | Edit | Protocol | # Used | URL |
|---|---|---|---|---|
| BitBucket | ... | OAuth1 | 4 | https://bitbucket.org/api/1.0/oauth/ |
| BitBucket (1) | ... | OAuth1 | 0 | https://bitbucket.org/api/1.0/oauth/ |
| BitBucket (2) | ... | OAuth1 | 0 | https://bitbucket.org/api/1.0/oauth/ |
| GitHub | ... | OAuth2 | 5 | https://github.com/login/oauth/a |
| BitBucket | ... | OAuth2 | 5 | https://bitbucket.org/site/oauth2/a |

The dialog has the following columns:

- *Name:* The name that was assigned to the settings definition when it was created[332]. The name cannot be edited.

- *Edit:* The button opens the OAuth Settings dialog[332], in which you can edit the settings of the selected definition.
- *Protocol:* Whether the definition uses OAuth1 or OAuth2.
- *# Used:* Refers to the number of times the definition is used in the current project (design).
- *URL*: The longest common part of the URLs of the definition's endpoints[332].

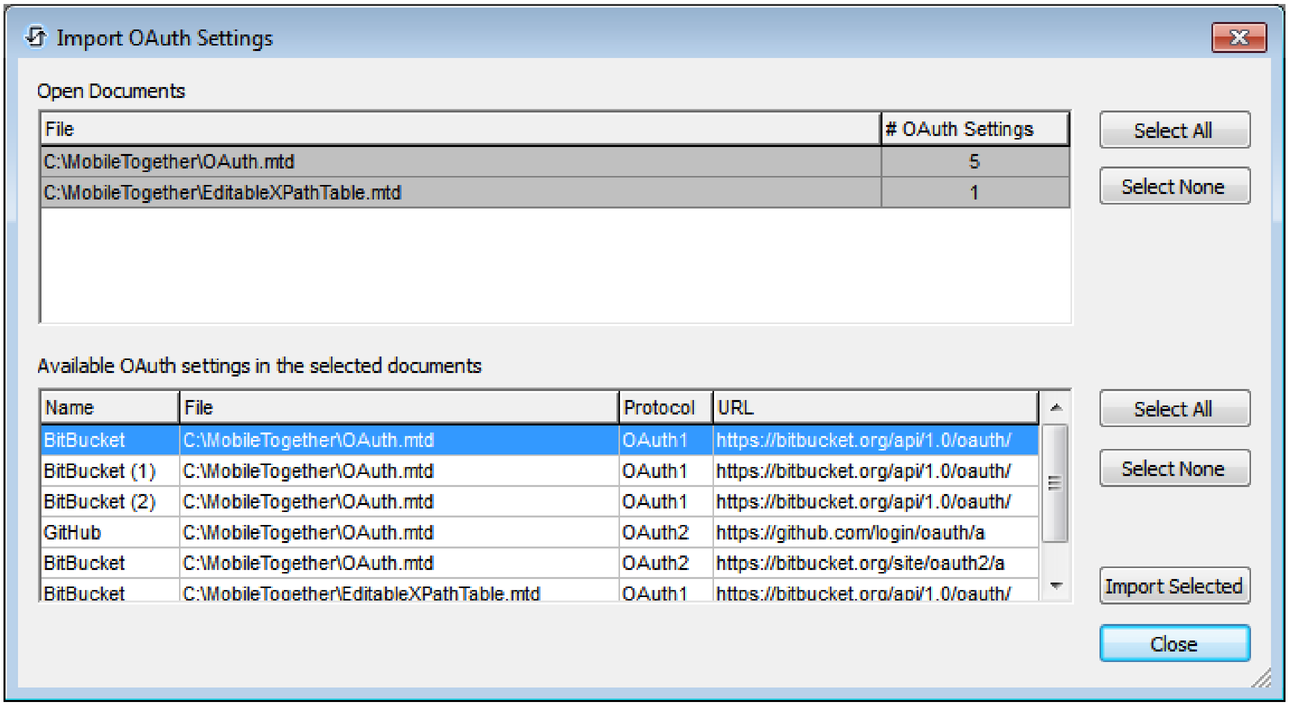
You can carry out the following actions in this dialog:

- *Delete Selected:* One or more definitions can be selected for deletion.
- *Select All*: Selects all the definitions.
- *Select None*: Selects none of the definitions.
- *Import:* Opens the **Import OAuth Settings**[1607] dialog, which enables you to import one or more OAuth definitions from other open MobileTogether Designer files. See the **Import OAuth Settings**[1607] dialog for details.

## 28.3.12 Import OAuth Settings

REST requests made in MobileTogether Designer can be authenticated with OAuth. See the section REST Request Settings[332] for a description of how to do this.

You can create multiple OAuth setting definitions in an MobileTogether Designer project. These are stored in a pool, and you can use a definition from the pool for authenticating REST requests defined anywhere in the document. The Import OAuth Settings dialog (*screenshot below*) enables you to import definitions from other open MobileTogether Designer projects into the current project.



Open Documents

| File | # OAuth Settings |
|---|---|
| C:\MobileTogether\OAuth.mtd | 5 |
| C:\MobileTogether\EditableXPathTable.mtd | 1 |

Available OAuth settings in the selected documents

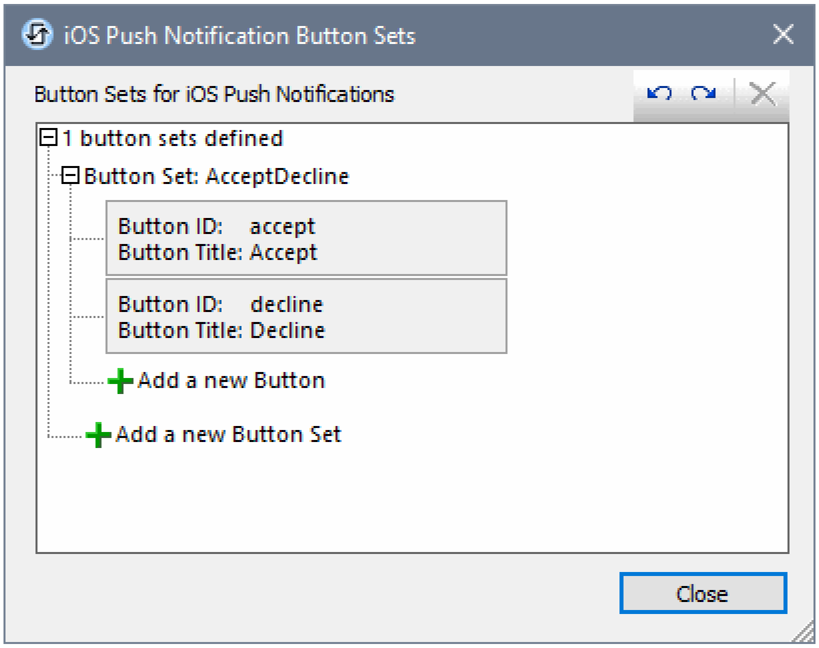
| Name | File | Protocol | URL |
|---|---|---|---|
| BitBucket | C:\MobileTogether\OAuth.mtd | OAuth1 | https://bitbucket.org/api/1.0/oauth/ |
| BitBucket (1) | C:\MobileTogether\OAuth.mtd | OAuth1 | https://bitbucket.org/api/1.0/oauth/ |
| BitBucket (2) | C:\MobileTogether\OAuth.mtd | OAuth1 | https://bitbucket.org/api/1.0/oauth/ |
| GitHub | C:\MobileTogether\OAuth.mtd | OAuth2 | https://github.com/login/oauth/a |
| BitBucket | C:\MobileTogether\OAuth.mtd | OAuth2 | https://bitbucket.org/site/oauth2/a |
| BitBucket | C:\MobileTogether\EditableXPathTable.mtd | OAuth1 | https://bitbucket.org/api/1.0/oauth/ |

The Open Documents pane (*see screenshot above*) displays all the other documents that are currently open in MobileTogether Designer. Select one or more documents to display their OAuth settings definitions in the lower pane. In the lower pane, select one or more definitions that you want to import into the current MobileTogether Designer project, and then click **Import Selected**. The definitions will be imported and can be viewed in the Maintain OAuth Settings[1606] dialog.

## 28.3.13 iOS Push Notification Button Sets

This command is used to create the buttons that appear in push notifications (PNs) received on iOS devices. iOS PN buttons are available in AppStore Apps[1471], not in standard MobileTogether apps. The PN button sets are created in the receiving solution. The button-set names defined via this command are used in the sending solution to indicate which button set to display[753] in the PN when the PN is received in the receiving solution. If the sending and receiving solution are the same, then the PN button sets that have been defined in the solution will be available in a combo box for selecting the PN's button set[753].

On selecting this command, the iOS Push Notification Button Sets dialog is displayed *(screenshot below)*.



Buttons are created in button sets of one, two, or three buttons. You can create as many button sets as you like; each button set can have a maximum of three buttons.

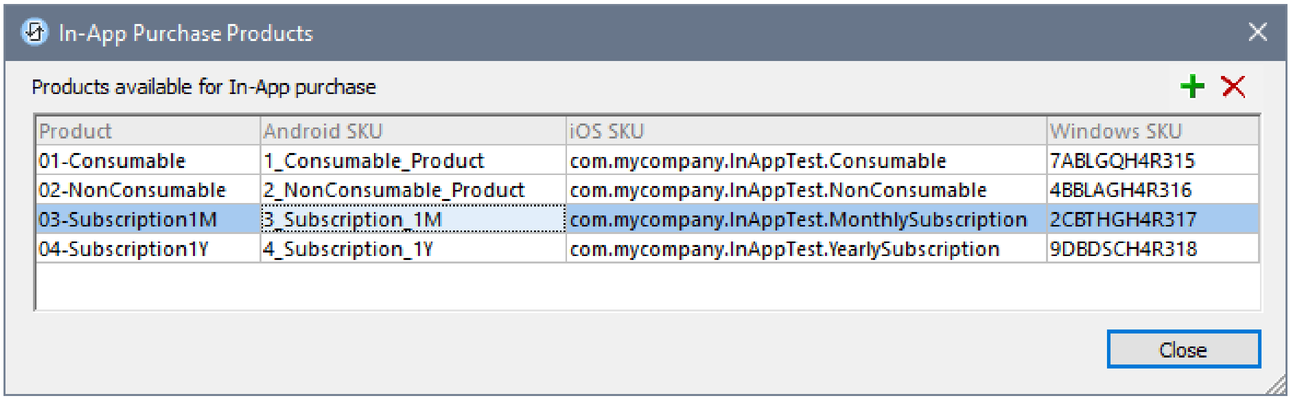
To create a button set, click *Add a new Button Set*. To add a new button to the set, click *Add a new Button*. To delete a button set or button, select the item and click the **Delete** icon in the top right-hand corner of the dialog. You can also undo and redo dialog-editing actions via icons in the top right of the dialog.

**Note:** PN buttons are not related in any way to Button controls[417].

See Push Notifications[1125] for an overview and more information about PNs.

## 28.3.14 In-App Purchase Products

This command opens the In-App Purchase Products dialog (*screenshot below*). Here you can map a product name (in the *Product* column) to the SKU IDs of the product in the respective app stores (Google, Apple, Windows). (These IDs are assigned when you create/register the product in your app store account.) In the MT design, each product is identified by the name you give it in this dialog. When the respective AppStore Apps[1471] are generated from the design, the appropriate SKU ID is used for the app on each platform.



| Product | Android SKU | iOS SKU | Windows SKU |
|---|---|---|---|
| 01-Consumable | 1_Consumable_Product | com.mycompany.InAppTest.Consumable | 7ABLGQH4R315 |
| 02-NonConsumable | 2_NonConsumable_Product | com.mycompany.InAppTest.NonConsumable | 4BBLAGH4R316 |
| 03-Subscription1M | 3_Subscription_1M | com.mycompany.InAppTest.MonthlySubscription | 2CBTHGH4R317 |
| 04-Subscription1Y | 4_Subscription_1Y | com.mycompany.InAppTest.YearlySubscription | 9DBDSCH4R318 |

To add a new mapping, click the **Plus** icon. To remove a mapping, click the **Delete** icon.

See the topic Register Products[1505] for details.

## 28.4 Refactor

The **Refactor** menu contains commands that list usages of various design components or  that relate to server action libraries and subprojects. It contains the following commands:

- List Usages of All Global Variables 1610
- List Usages of All Page Source Variables 1610
- List Page Sources by Attribute 1610
- List Usages of All User-Defined XPath/XQuery Functions 1611
- List Usages of All Action Groups 1611
- List Usages of All Style Sheets 1612
- List All File and Directory References 1612
- List All External Data References 1612
- List Unused Functions, User Variables, Etc 1613
- Replace DB Sources 1613
- Add Server Action Library 1614
- Open Server Action Library Individually 1614
- Extract New Subproject 1614
- Include Subproject 1616
- Open Subproject Individually 1616

### 28.4.1 List Usages of All Global Variables

Returns a list, in the Listings Pane 281, of all global variables 1298. Each global variable is listed together with information about where the variable is used. This listing contains links that take you directly to the definition containing the usage, enabling you to quickly locate and edit that definition.

### 28.4.2 List Usages of All Page Source Variables

Returns a list, in the Listings Pane 281, of all page source variables 349, each of which indicates a page source. Each page source variable contains links to the definitions in which that variable is used. This enables you to quickly locate and edit that definition. Clicking the page source link itself highlights the page source in the Page Sources Pane 270.

### 28.4.3 List Page Sources by Attributes

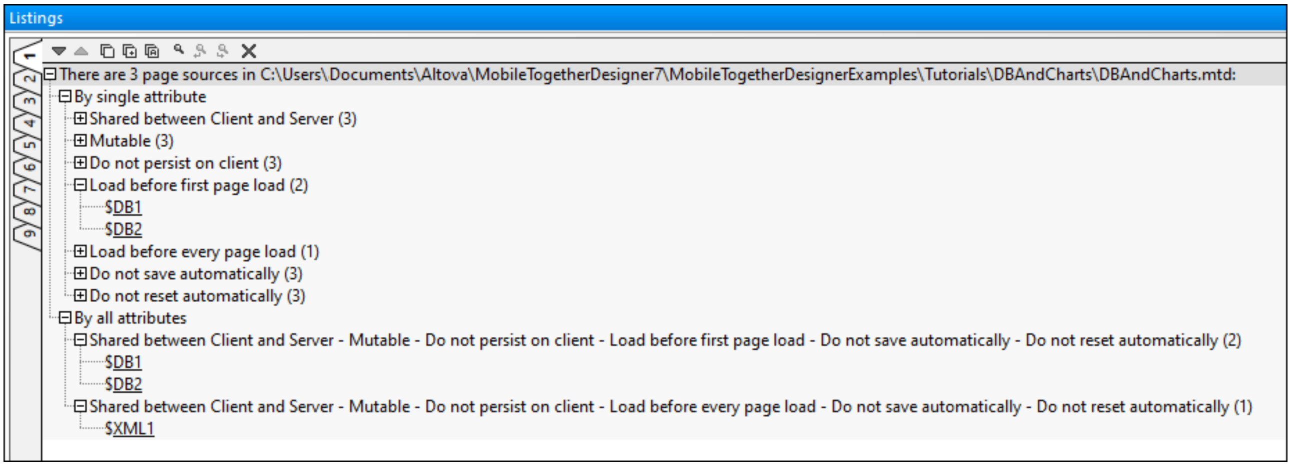
Returns a list, in the Listings Pane 281, of the current project's page sources 349, organized by page source attributes (that is, according to the values of the project's various settings). Two lists are returned.

*List by single attribute (value)*
If an attribute value has been set for any of the project's page sources, the attribute is listed together with the page sources that have this value set. The following page source attributes are evaluated:

- Client only, Server only, or Shared
- Read-only or Editable (Mutable)
- Persist on client or not
- Load behavior
- Save behavior
- Reset Behavior

*List by coincidence of values of all attributes*

The page sources are grouped together if they have the same values for all attributes. See screenshot below.



```
⊟There are 3 page sources in C:\Users\Documents\Altova\MobileTogetherDesigner7\MobileTogetherDesignerExamples\Tutorials\DBAndCharts\DBAndCharts.mtd:
  ⊟By single attribute
    ⊞Shared between Client and Server (3)
    ⊞Mutable (3)
    ⊞Do not persist on client (3)
    ⊟Load before first page load (2)
        $DB1
        $DB2
    ⊞Load before every page load (1)
    ⊞Do not save automatically (3)
    ⊞Do not reset automatically (3)
  ⊟By all attributes
    ⊟Shared between Client and Server - Mutable - Do not persist on client - Load before first page load - Do not save automatically - Do not reset automatically (2)
        $DB1
        $DB2
    ⊟Shared between Client and Server - Mutable - Do not persist on client - Load before every page load - Do not save automatically - Do not reset automatically (1)
        $XML1
```

Clicking a page source link highlights the page source in the Page Sources Pane[270].

## 28.4.4 List Usages of All User-Defined XPath/XQuery Functions

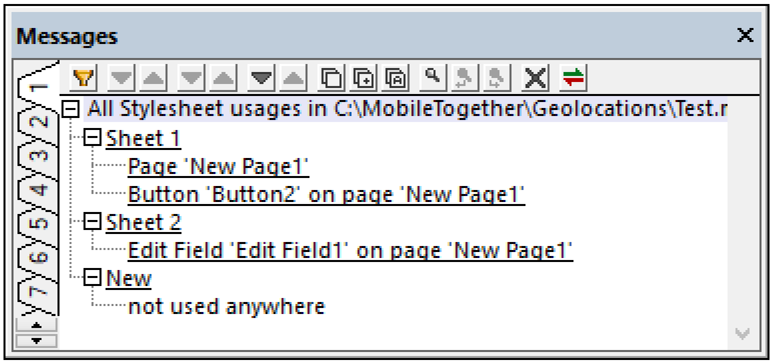
Returns a list, in the Listings Pane[281], of all the user-defined XPath/XQuery functions used by pages in the project, together with the pages in which they occur. Clicking the links in the listing takes you directly to the dialog defining the XPath/XQuery function or the definition containing the XPath/XQuery function.

## 28.4.5 List Usages of All Action Groups

Returns a list, in the Listings Pane[281], of all the Action Groups used by pages in the project. The list is ordered by Action Group. Each Action Group is sub-divided into direct and indirect usages. Pages that use the Action Group are listed together with the event that triggers the Action Group. Clicking the links in the listing takes you directly to the dialog defining the action group or the event for which the action group is defined.

## 28.4.6 List Usages of All Style Sheets

A user-created style sheet[1318] can be applied to page instances, table instances, and control instances. This command returns a list, in the Listings Pane[281], of all the user-created style sheets[1318] in the project and where each is used (*see screenshot below*).



All Stylesheet usages in C:\MobileTogether\Geolocations\Test.r
- Sheet 1
  - Page 'New Page1'
  - Button 'Button2' on page 'New Page1'
- Sheet 2
  - Edit Field 'Edit Field1' on page 'New Page1'
- New
  - not used anywhere

The list is ordered by style sheet. Under each style sheet, all the page, table, and control instances are listed that use that style sheet. Clicking one of these items takes you directly to the respective page, table, or control.

If a style sheet is not used, then this is reported (*see screenshot above*). Unused style sheets are also reported via  the **List Unused Functions, User Variables, Style Sheets, Action Groups**[1613] command of the **Project** menu.

## 28.4.7 List All File and Directory References

Returns a list, in the Listings Pane[281], of all the files and directories that are referenced in the project. The list also includes controls that reference file sources, even if the reference is via an XPath expression. For example, an image control references an image file, so the image will be included in the list. Clicking the links in the listing takes you directly to the design definition that references the selected file or directory.

**Note:** The database files of file-based databases, such as MS Access and SQLite, are not listed.

## 28.4.8 List All External Data References

Returns a list, in the Listings Pane[281], of all the external data sources that are referenced in the project. These include data sources accessed via HTTP, REST, and SOAP. Clicking the links in the listing takes you directly to the design definition that references the selected resource.

## 28.4.9 List Unused Functions, Variables, Etc

Returns a list, in the Listings Pane[281], of all the **unused** XPath/XQuery functions[1591], user variables[1309], page sources[315], pages[257], style sheets[1318], localized strings[1600], and action groups[1594] that are defined in the project. This is useful if you wish to review these user-defined components that are not in use and clean up the project. Clicking the links in the listing takes you directly to the respective definition.

*Unused localized strings*
A **used** localized string is one that is submitted as an argument of the mt-load-string function[1262].

An unused localized string, on the other hand, is one that is not used with the mt-load-string function[1262], either directly or indirectly.

- Without any potential usage: The string name is not used as the argument of the mt-load-string function[1262].
- With potential usage: The string name is used indirectly in an XPath expression involving the mt-load-string function[1262]. For example:

```
let $v := 'UnusedStringName' return mt-load-string($v)
let $v := 'UnusedStringName' return $v => mt-load-string()
'UsedStringName' => mt-load-string() || 'UnusedStringName'
```

In the last example above, although `UnusedStringName` is not submitted to `mt-load-string()`, it is listed with potential usage.

## 28.4.10 Replace DB Sources

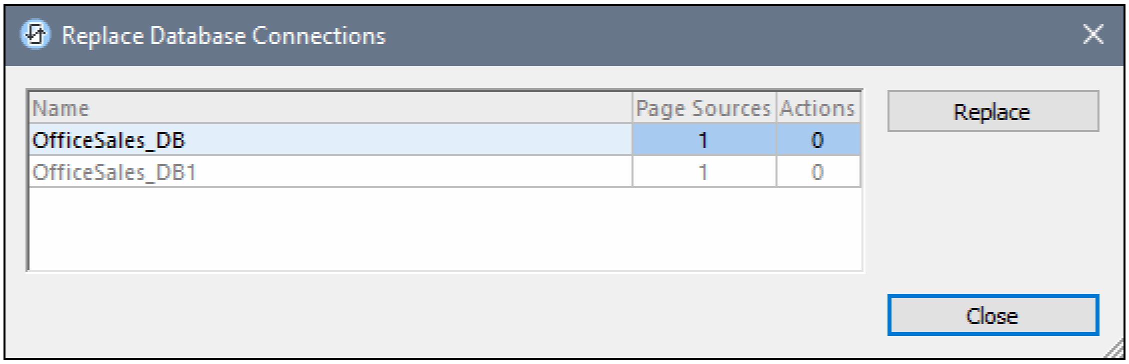
The **Replace DB Sources** command enables you to switch the database connection of a DB page source. This can be useful, for example, if you wish to switch from a test environment to a production environment. In order for the DB switch to be successful, the two DBs must be identical in type and structure, and the following conditions must be met:

- The type of the original and replacement DBs must be the same. For example, both DBs would be MS SQL Server.
- The name and structure of the root object of both DBs must be the same.
- The structure of the selected objects in both DBs must be the same.

**Note:** If the root object is not the same or if the structures of objects do not match, then no error or warning is reported. We therefore recommend that you check whether the switch has been successfully implemented.

### How to replace a DB connection

When the command is executed, the Replace Database Connections dialog appears (*see screenshot below*). The dialog lists the DB connections that are currently active in the design, with each connection showing the number of page sources and the number of actions that use that DB connection.



| Name | Page Sources | Actions |
|---|---|---|
| OfficeSales_DB | 1 | 0 |
| OfficeSales_DB1 | 1 | 0 |

To replace one of the listed DB connections, select it and click **Replace**. This opens the Database Connection Wizard[961], with the help of which you can define the replacement connection that you want.

## 28.4.11 Add Server Action Library

This command, which is also available in the context menu of the Files Pane[259], opens a File Open dialog box. Here you can browse for the server action library file[1551] that you want to add to the current project. After selecting a server action library file in the Open dialog and clicking **Open**, the file will be added as a server action library and will appear as such in the current project's Files Pane[259].

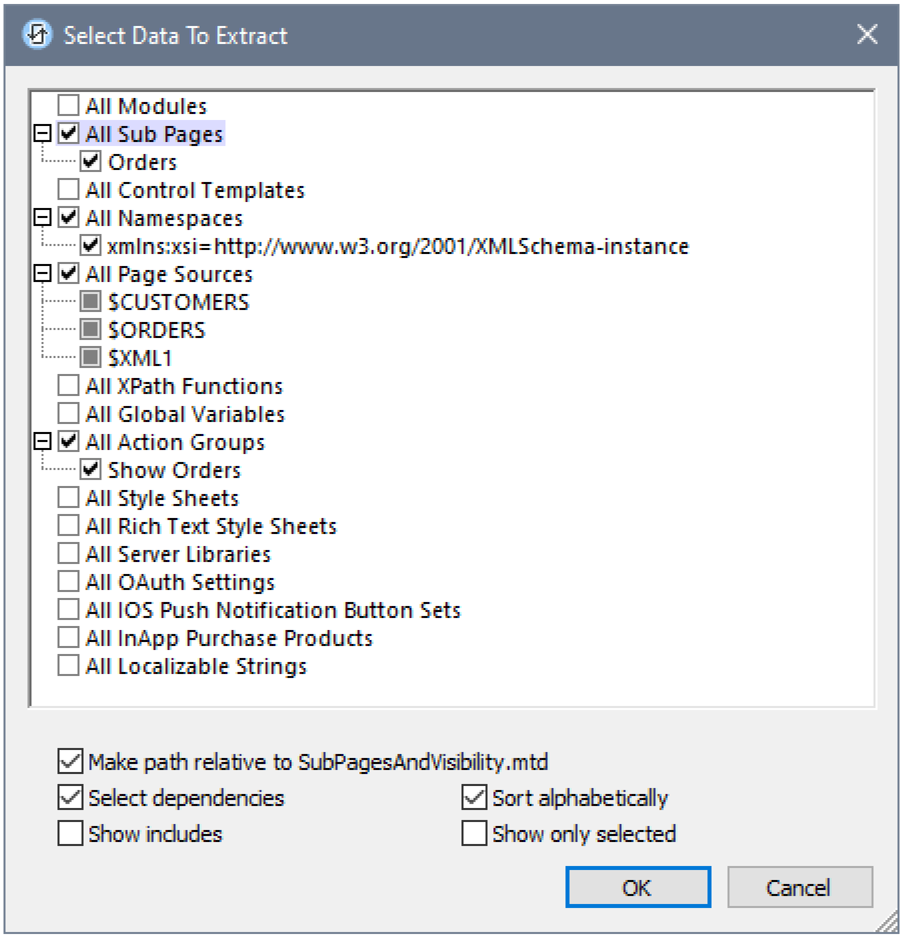
## 28.4.12 Open Server Action Library Individually

This command enables you to open server action libraries[1551] of the current project in separate tabs of MobileTogether Designer. On hovering over this command, a submenu appears that contains a list of the server action libraries of the current project. Select the server action library you want to open. If no server action library exists, then no submenu rolls out.

Server action libraries of the current project are listed in the latter's Files Pane[259].

## 28.4.13 Extract New Subproject

This command enables you to extract components of the current project to a new subproject file (which, similar to a standard design file, has a `.mtd` file extension). Clicking this command brings up the Select Data to Extract dialog (*screenshot below*), which displays the current project's components. Select the components you want to extract to the subproject.



Select Data To Extract

- □ All Modules
- ☑ All Sub Pages
  - ☑ Orders
- □ All Control Templates
- ☑ All Namespaces
  - ☑ xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
- ☑ All Page Sources
  - ■ $CUSTOMERS
  - ■ $ORDERS
  - ■ $XML1
- □ All XPath Functions
- □ All Global Variables
- ☑ All Action Groups
  - ☑ Show Orders
- □ All Style Sheets
- □ All Rich Text Style Sheets
- □ All Server Libraries
- □ All OAuth Settings
- □ All IOS Push Notification Button Sets
- □ All InApp Purchase Products
- □ All Localizable Strings

☑ Make path relative to SubPagesAndVisibility.mtd

☑ Select dependencies ☑ Sort alphabetically

□ Show includes □ Show only selected

Note the following points:

- If a selected component uses other components and the *Select Dependencies* option is checked (*see screenshot*), then all dependent components will also be selected.
- If you click a component group that contains no item, then nothing happens. Such component groups will have no child items. In the screenshot above, for example, the component group *All Namespaces* has one item, whereas the component group *All Style Sheets* has no item.
- The *Relative Path* option applies to any references to the subproject that are written and displayed in the current project.
- The *Select Dependencies* option can be checked at any time. On being selected, it will apply also to already selected items.
- The other dialog options are related to the display features of the dialog: (i) to sort components in each group alphabetically; (ii) to show components of included subprojects; and (iii) to show only the selected components. The last option is useful if the project contains several components. However, it should be switched off in order to see what other components than the selected components are available.

After you finish selecting the components to extract, click **OK**. In the Save As dialog that appears, browse for the location where you want to save the subproject and give the subproject a name. On clicking **Save**, the following happens:

- The subproject file will be saved at the specified location with a `.mtd` file extension.
- The subproject file will be referenced from the main project and will be listed in its Files Pane[259] as a subproject.
- The components that were extracted to the subproject file will be removed from the main project. They will now be accessed from the referenced subproject and are displayed in gray in the main project. If you want to copy (rather than include)[1350] the components of a subproject into a project, go to the Files Pane[259] and use its context menu command **Include Subproject as a Copy**.

**Note:** If you delete a subproject in the Files Pane[259] of a project, then references to the subproject's components will be lost. However, if the project is reloaded before the modified file is saved, you will be prompted for the location of the missing subproject files.

**Note:** If you want to include a subproject after having extracted, do so via the menu command **Refactor | Include Subproject** (the same command is also available in the context menu of the Files Pane[259]).

## 28.4.14 Include Subproject

This command, which is also available in the context menu of the Files Pane[259], opens a File Open dialog box, in which you can browse for the MobileTogether Designer project file (`.mtd` file) that you want to add as a subproject of the current project. After selecting a server action library file and clicking **Open** in the Open dialog, the file will be added as a subproject and will appear as such in the current project's Files Pane[259].

**Note:** If the file you want to add as a subproject contains a visible top-level page, then the file cannot be added as a subproject. You will receive an error message to this effect and a suggestion to assign all top-level pages in the project that you want to create as a subproject to a hidden module that will not be exported. For information about how to do this, see the topic Modules[1352].

## 28.4.15 Open Subproject Individually

This command opens subprojects[1349] of the current project in separate tabs of MobileTogether Designer. On hovering over this command, a submenu rolls out that contains a list of the subprojects of the current project. In this submenu, select the subproject you want to open. If no subproject exists, then no submenu rolls out.

Subprojects of the current project are listed in the latter's Files Pane[259].

## 28.5 Run

The **Run** menu contains commands to run simulations 1355 and automated-testing procedures 1399. It contains the following commands:

- Simulate Workflow 1617
- Trial Run on Client 1618
- Use Server for Workflow Simulation 1619
- Simulate Server Deployment 1620
- Simulation Options 1620
- Record New Test Case 1623
- Playback Test Case 1623
- Trial Run Test Cases on Client 1623
- Manage Test Cases and Runs 1624
- Run RecordsManager 1625

### 28.5.1 Simulate Workflow

⊟ *Icon*

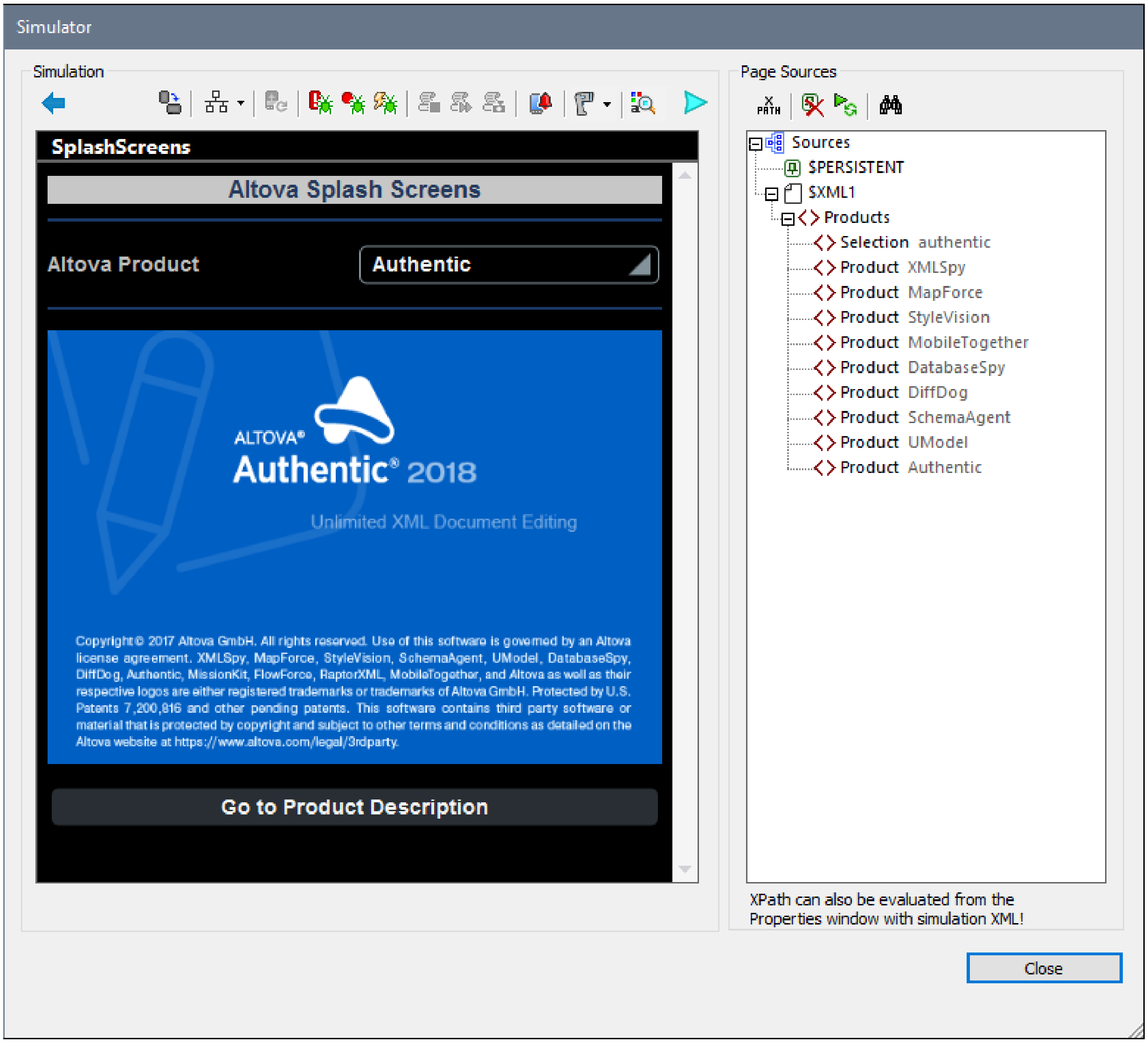
▶

⊟ *Shortcut*

**F5**

⊟ *Description*

Starts the local (MobileTogether Designer) simulator in a separate window for testing purposes (*see screenshot below*). The simulator goes step-by-step through the workflow of the currently active project. The mobile client that is currently selected in the Device Selector of the Page Settings toolbar 253 will be simulated.



Altova Splash Screens

Altova Product: Authentic

Copyright © 2017 Altova GmbH. All rights reserved. Use of this software is governed by an Altova license agreement. XMLSpy, MapForce, StyleVision, SchemaAgent, UModel, DatabaseSpy, DiffDog, Authentic, MissionKit, FlowForce, RaptorXML, MobileTogether, and Altova as well as their respective logos are either registered trademarks or trademarks of Altova GmbH. Protected by U.S. Patents 7,200,816 and other pending patents. This software contains third party software or material that is protected by copyright and subject to other terms and conditions as detailed on the Altova website at https://www.altova.com/legal/3rdparty.

Go to Product Description

```
Sources
  $PERSISTENT
  $XML1
    Products
      Selection  authentic
      Product  XMLSpy
      Product  MapForce
      Product  StyleVision
      Product  MobileTogether
      Product  DatabaseSpy
      Product  DiffDog
      Product  SchemaAgent
      Product  UModel
      Product  Authentic
```

XPath can also be evaluated from the Properties window with simulation XML!

## 28.5.2 Trial Run on Client

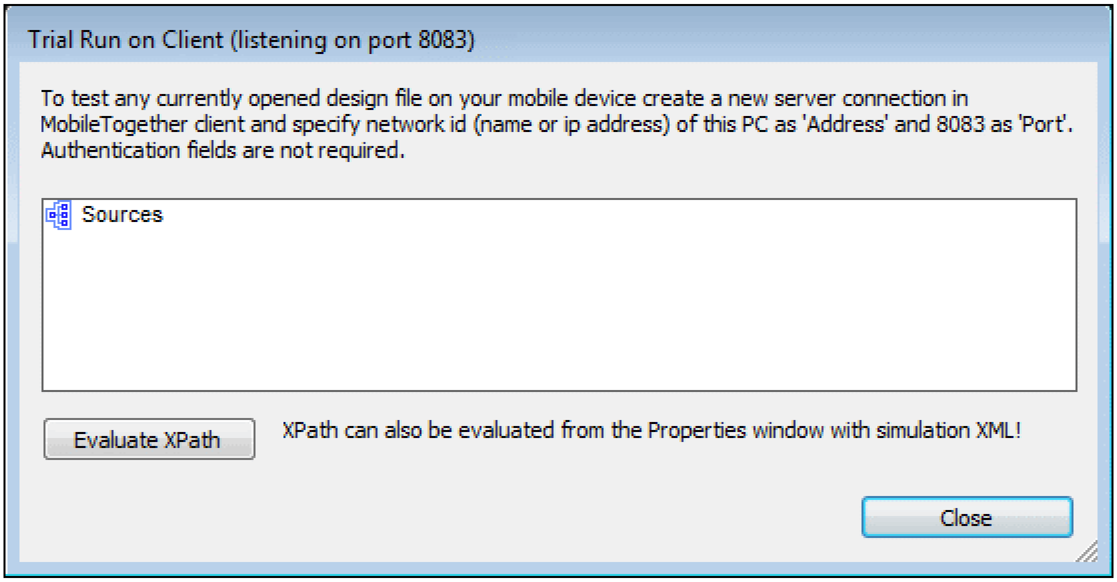
⊟ *Icon*

⊟ *Shortcut*

**Shift+F5**

⊟ Description

Tests the active MobileTogether design file on the specified client. MobileTogether Designer itself acts as the MobileTogether Server and serves the design and related data files directly to the client. In the MobileTogether Client app on your mobile device, you must set up a server connection to the local PC running MobileTogether Designer. Note that, by default, **8083** is the port on your local PC to which the client must connect. You can change this port in the Trial Run on Client tab[1663] of the Options dialog. After the connection between client and PC is established and the design is selected on the client, the Sources tree in the Trial Run on Client dialog (*screenshot below*) is populated and the trial run (simulation) begins.


Trial Run on Client (listening on port 8083)

To test any currently opened design file on your mobile device create a new server connection in MobileTogether client and specify network id (name or ip address) of this PC as 'Address' and 8083 as 'Port'. Authentication fields are not required.

XPath can also be evaluated from the Properties window with simulation XML!

## 28.5.3 Use Server for Workflow Simulation

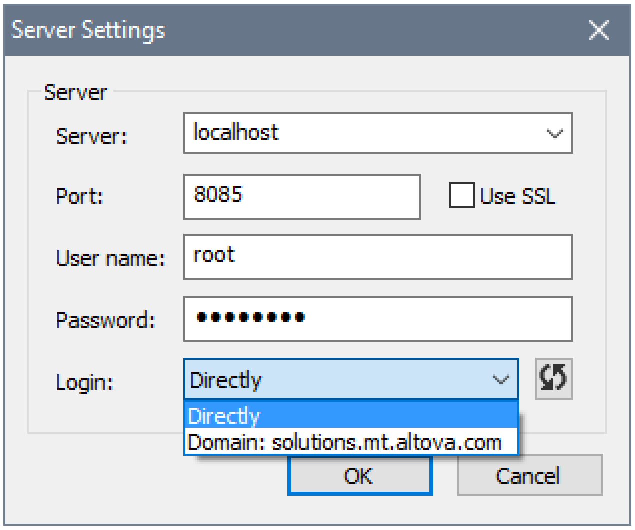
⊟ Icon

⊟ Shortcut

**Ctrl+F5**

⊟ Description

Displays the Server Settings dialog (*screenshot below*). Enter the connection and authentication details of the MobileTogether Server on which you want to run the simulation. You can log in directly, or via a domain login[1663] if this has been set up (*see the description of the Server Settings tab of the Options dialog*[1663]). On clicking **OK**, the simulation starts in a separate window.



Server Settings

Server

Server: localhost

Port: 8085 □ Use SSL

User name: root

Password: ●●●●●●●●●

Login: Directly

Directly

Domain: solutions.mt.altova.com

## 28.5.4 Simulate Server Deployment

Run this command to simulate the OnServerDepoyment actions[296] if these have been defined. These actions are defined in the OnServerDeployment actions tree[291], which you can access via the More Project Settings dialog of a project[296].

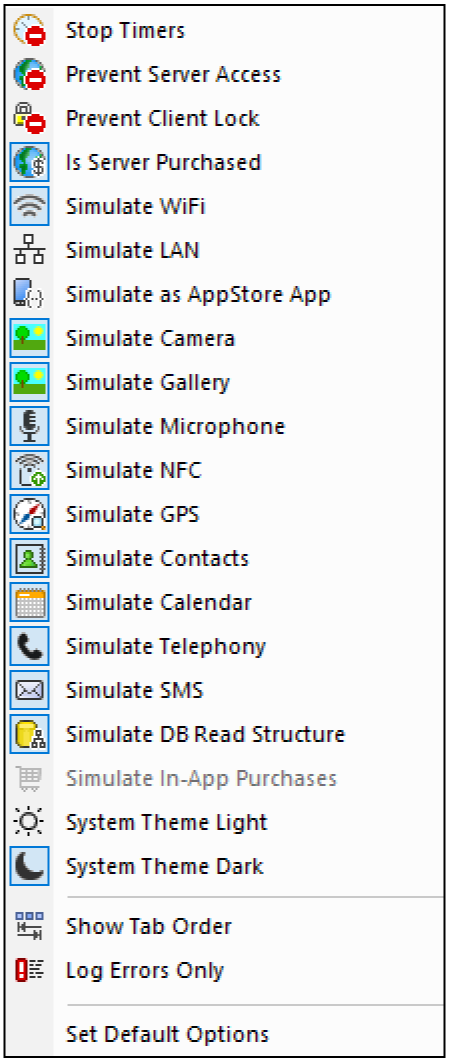
If no server deployment actions have been defined, then a message box will be displayed that informs you about these actions and where they can be defined.

More more information about server deployment actions, see the topic Deploying the Project[291].

## 28.5.5 Simulation Options

Hover this command to display a submenu in which you can select the options you want to apply for simulations. If a simulation is already running, then a selection is immediately applied to the running simulation. If a simulation is not currently running, then selections will be applied for all subsequent simulations till an option is deselected.

The following simulation options are available:



- *Stop Timers:* If a timer has been set to run at intervals 390 and actions have been defined to be executed at these intervals, you can stop timers (and, consequently, actions) by clicking **Stop Timers**. This will clear up the clutter of messages generated by these actions, and will allow you to more easily analyze other messages and aspects of the workflow.
- *Prevent Server Access:* When selected, disables access to the server, and so allows you to test the solution's behavior in a server-connection-error scenario. When unselected, server access is allowed. For more information about this feature, see Server Connection Errors 394.
- *Prevent Client Lock:* When selected, prevents the client being locked from server access when the Lock Client 919 action is executed. If the server is inaccessible because the Lock Client 919 action has been executed from another client, then preventing the lock will of course not work.
- *Is Server Purchased:* For simulations in the designer and for trial runs on client, simulates that MobileTogether Server licenses have been purchased. For simulations on the server, the actual purchase-state of licenses on the server is returned.
- *Simulate WiFi:* Sets the `mt-connected-via-wifi` 1262 XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This allows the simulator to behave as though WiFi access is available. In this way, you can simulate design scenarios in which WiFi access is required.
- *Simulate LAN:* Sets the `mt-connected-via-lan` 1262 XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This allows the simulator to behave as though a LAN connection

is available. As a result, you can simulate design scenarios that require a LAN connection.

- *Simulate as AppStore App:* Sets the static global variable **`MT_IsAppStoreApp`** 1300 to `true()` when enabled, to `false()` when disabled. This allows simulations to be carried out that are conditional on the value of this variable.
- *Simulate Camera:* When toggled on, the simulator behaves as though the device's camera is available. This enables you to simulate design scenarios that require camera access.
- *Simulate Gallery:* When toggled on, the simulator behaves as though the device's photo gallery is available. This enables you to simulate design scenarios that require gallery access.
- *Simulate Microphone:* When toggled on, the simulator behaves as though the device's microphone is available. This enables you to simulate design scenarios that require microphone access.
- *Simulate NFC:* When selected, enables NFC functionality so that NFC actions can be executed. Actual NFC data is supplied to the simulator via NFC sample files 1377.
- *Simulate Bluetooth (BT):* When toggled on, the simulator behaves as though the device's Bluetooth connectivity is available. This enables you to simulate Barcode Scanner scenarios 1145 that require Bluetooth.
- *Simulate GPS:* When selected, enables geolocation functionality so that geolocation features can be tested. Dummy geolocations can be supplied via the Geolocations XML file 1372, which is used specifically to supply geolocations for simulations.
- *Simulate Contacts:* When toggled on, the simulator behaves as though the device's address book is available. This enables you to simulate design scenarios that require access to the address book. The address book is simulated either from a sample file 1381 or from your Microsoft Outlook contacts 1663. Which option to use is specified in the Simulation 2 tab of the Options dialog 1669.
- *Simulate Calendar:* When toggled on, the simulator behaves as though the device's calendar is available. This enables you to simulate design scenarios that require access to the calendar. The calendar is simulated either from a sample file 1382 or from your Microsoft Outlook calendar 1663. Which option to use is specified in the Simulation 2 tab of the Options dialog 1669.
- *Simulate Telephony:* When toggled on, the simulator behaves as though the device's telephony functionality is available. This enables you to simulate design scenarios that require telephone access.
- *Simulate SMS:* When toggled on, the simulator behaves as though the device's SMS functionality is available. This enables you to simulate design scenarios that require SMS access.
- *Simulate DB Read Structure:* When toggled on, the simulator takes the DB structure from the XML file that is specified in the Simulation 2 tab of the Options dialog 1669. For relevant information, see the DB Read Structure 867 action.
- *Simulate Telephony:* When toggled on, the simulator behaves as though the device's telephony functionality is available. This enables you to simulate design scenarios that require telephone access.
- *Simulate In-App Purchases:* When selected, enables the simulation of In-App Purchases 1502 by using sample data that is stored in an XML file 1516. The XML file to use is specified in the Simulation 2 tab of the Options dialog 1669.
- *System Theme Light/Dark:* Switches to the selected theme (light or dark).
- *Show Tab Order:* If tab ordering 1633 has been set, then select this option to show all tabbed controls with their respective tab-order number.
- *Log Errors Only:* Select this option to log errors only and to ignore other types of messages.
- *Set Default Options:* Resets Simulation pane options to their default settings 1663.

## 28.5.6 Record New Test Case

⊟ *Icon*

⊟ *Description*

Starts a new test case in the Simulator[1355] and records user actions. When recording stops, you are prompted to give the recording a name and save it as a test case. Recording options are specified in the *Manage Test Cases and Runs*[1401] dialog. *See Recording a Test Case*[1401] *for details*.

## 28.5.7 Playback Test Case

⊟ *Icon*

⊟ *Description*

Plays back the test case that is selected in the *Available Test Cases for Playback* combo box. If the playback returns differences from the test case, then the playback is saved. *See Playing Back a Test Case*[1403] *for details*. Note that options for playback are specified in the *Manage Test Cases and Runs*[1401] dialog.

## 28.5.8 Trial Run Test Cases on Client

⊟ *Icon*

⊟ *Shortcut*

**Alt+F5**

⊟ *Description*

Plays back, on a connected client, the test case that is selected in the *Available Test Cases for Playback* combo box. If the playback returns differences, then the playback is saved. *See Playing Back a Test Case*[1403] *for details*.

## 28.5.9 Manage Test Cases and Runs

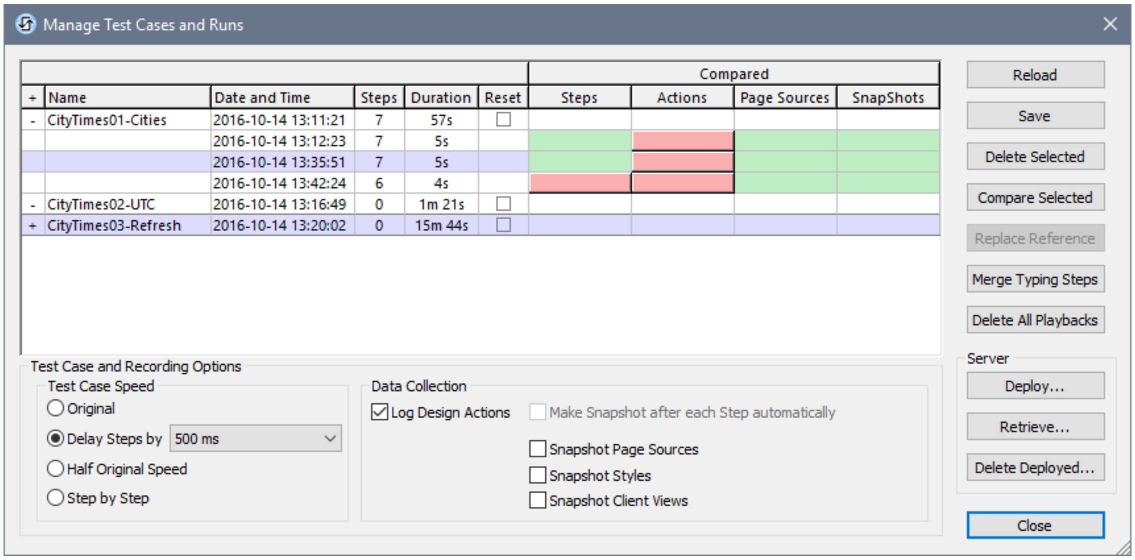
⊟ *Icon*

⊟ *Description*

Displays the *Manage Test Cases and Runs* 1401 dialog.

In the Manage Test Cases and Runs dialog (*screenshot below*) you can do the following:

- Set up recording options for test cases.
- Set up recording and playback options for subsequent test runs.
- Load and save MobileTogether Recording files (`.mtrecord` files).
- Delete and compare test runs.
- Substitute a test case with one of its test runs. The test run takes on the role of the test case. Other test runs are deleted, and the selected test run becomes the new test case of that (now empty) group.
- Deploy a test case to MobileTogether Server, retrieve test runs from the server, and delete a test case or test run from the server.



| | | | | | | Compared | | | |
|---|---|---|---|---|---|---|---|---|---|
| + | Name | Date and Time | Steps | Duration | Reset | Steps | Actions | Page Sources | SnapShots |
| - | CityTimes01-Cities | 2016-10-14 13:11:21 | 7 | 57s | □ | | | | |
| | | 2016-10-14 13:12:23 | 7 | 5s | | 🟩 | 🟥 | 🟩 | 🟩 |
| | | 2016-10-14 13:35:51 | 7 | 5s | | 🟩 | 🟥 | 🟩 | 🟩 |
| | | 2016-10-14 13:42:24 | 6 | 4s | | 🟥 | 🟥 | 🟩 | 🟩 |
| - | CityTimes02-UTC | 2016-10-14 13:16:49 | 0 | 1m 21s | □ | | | | |
| + | CityTimes03-Refresh | 2016-10-14 13:20:02 | 0 | 15m 44s | □ | | | | |

Test Case and Recording Options

Test Case Speed
○ Original
● Delay Steps by 500 ms
○ Half Original Speed
○ Step by Step

Data Collection
☑ Log Design Actions
□ Make Snapshot after each Step automatically
□ Snapshot Page Sources
□ Snapshot Styles
□ Snapshot Client Views

*See Managing Test Cases and Runs* 1406 *for details*.

## 28.5.10 Run RecordsManager

⊟ *Icon*

⊟ *Description*

RecordsManager[70] is available in your installation of MobileTogether Designer (version 8.0 onwards) for simulation and trying out. The menu command **Run | Run RecordsManager**[1625] starts a simulation of RecordsManager in MobileTogether Designer. Note that you will not be able to edit the design or to deploy the solution to the server.

You can read more about RecordsManager at the Altova website.

## 28.6 Debug

The **Debug** menu contains commands that apply to MT Debugger and the debugging of XPath expressions and actions. It contains the following commands:

- Resume Debugging / Go[1626]
- Stop Debugging[1626]
- Step into Action[1627]
- Step into XPath[1627]
- Step Out[1628]
- Step Over[1628]
- Run to Selected Action[1628]
- Stop at Next Error[1629]
- Stop at Next Breakpoint[1629]
- Stop at Next Action[1629]
- Disable All Breakpoints[1630]
- Enable All Breakpoints[1630]
- Remove All Breakpoints[1630]
- Debug Windows[1630]

### 28.6.1 Resume Debugging / Go

⊟ *Icon*

▶

⊟ *Shortcut*

**F5**

⊟ *Description*

Starts the Actions Debugger[1390] or XPath Debugger[1252], whichever is active. It can also be used to resume debugging where this has been paused. The command is enabled only when one of these debuggers has been opened from a simulation. See MT Debugger[1388] for more information.

### 28.6.2 Stop Debugging

⊟ *Icon*

■

⊟ *Shortcut*

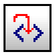
**Shift+F5**

*Description*

Stops debugging and closes MT Debugger[1388]. It is enabled when MT Debugger[1388] has been started, that is, if (i) a debugger mode[1389] has been selected, and (ii) a simulation has been started.

## 28.6.3 Step into Action

*Icon*



*Shortcut*

**F11**

*Description*

Proceeds through the action execution in Actions Debugger[1390], one step at a time. The command is enabled after the Actions Debugger[1390] has been opened.

## 28.6.4 Step into XPath

*Icon*

*Shortcut*

**Ctrl+Shift+F11**

*Description*

Opens the XPath Debugger[1398] and displays the XPath expression of the action. The command is enabled after the Actions Debugger[1390] has been opened.

## 28.6.5 Step Out

⊟ *Icon*

⊟ *Shortcut*

**Shift+F11**

⊟ *Description*

Steps out of the current action execution step, and goes to the parent step. The command is enabled after the Actions Debugger[1390] has been opened.

## 28.6.6 Step Over

⊟ *Icon*

⊟ *Shortcut*

**Ctrl+F11**

⊟ *Description*

Steps over descendant steps of the action execution. The command is enabled after the Actions Debugger[1390] has been opened.

## 28.6.7 Run to Selected Action

⊟ *Icon*

⊟ *Shortcut*

**Ctrl+F5**

⊟ *Description*

The command is enabled after the Actions Debugger[1390] has been opened. If the command is clicked, then the Debugger executes all actions up to the current cursor location and stops there. If a breakpoint is encountered at an earlier point, then the debugger stops at this point and can be resumed by running the **Run to Selected** command again. If the action at the current cursor location cannot reached during execution, then action execution proceeds without stopping.

## 28.6.8 Stop at Next Error

*Icon*

*Description*

Stops the simulation at the next XPath error and displays the XPath expression in XPath Debugger[1388]. This command enables you to specify the debugger mode[1389] of a simulation before you start it. You can also use this command to change the debugger mode after a simulation has been started.

## 28.6.9 Stop at Next Breakpoint

*Icon*

*Description*

Stops the simulation at the next breakpoint (which may be on an action or XPath expression) and opens the appropriate Debugger[1388]. This command enables you to specify the debugger mode[1389] of a simulation before you start it. You can also use this command to change the debugger mode after a simulation has been started.

## 28.6.10 Stop at Next Action

*Icon*

⊟ *Description*

Stops the simulation at the action/s of the next event that is triggered and displays the action/s in the Actions Debugger[1390]. This command enables you to specify the debugger mode[1389] of a simulation before you start it. You can also use this command to change the debugger mode after a simulation has been started.

## 28.6.11 Disable All Breakpoints

Breakpoints can be temporarily disabled, in which case they will be ignored during debugging. This command disables all breakpoints that have been set on actions[1390] and on XPath expressions[1398]. You can also disable all (or individual) breakpoints in the Breakpoints Pane[269]. For more information about breakpoints, see MT Debugger[1388].

## 28.6.12 Enable All Breakpoints

Breakpoints can be temporarily disabled, in which case they will be ignored during debugging. This command ensures that all breakpoints are enabled, including those that have been disabled. It applies to all breakpoints, whether set on actions[1390] or on XPath expressions[1398]. You can also enable all (or individual) breakpoints in the Breakpoints Pane[269]. For more information about breakpoints, see MT Debugger[1388].

## 28.6.13 Remove All Breakpoints

Remove all breakpoints in the design, whether set on actions[1390] or on XPath expressions[1398]. You can also remove all (or individual) breakpoints in the Breakpoints Pane[269]. For more information about breakpoints, see MT Debugger[1388].

## 28.6.14 Debug Windows

Placing the cursor over the **Debug Windows** causes a sub menu to appear, which has the following commands:

- *Breakpoints:* The Breakpoints Pane[269] is, by default, tabbed together with the Controls Pane[266]. Click this command to make the Breakpoints Pane[269] the active tab.
- *Pin Debugger Action View:* This command is enabled only when MT Debugger[1388] has been started, that is, if (i) a debugger mode[1389] has been selected, and (ii) a simulation has been started. It makes the Actions Debugger[1390] the active window.
- *Pin Debugger XPath View:* This command is enabled only when MT Debugger[1388] has been started, that is, if (i) a debugger mode[1389] has been selected, and (ii) a simulation has been started. It starts the XPath Debugger[1398] and makes it the active window.

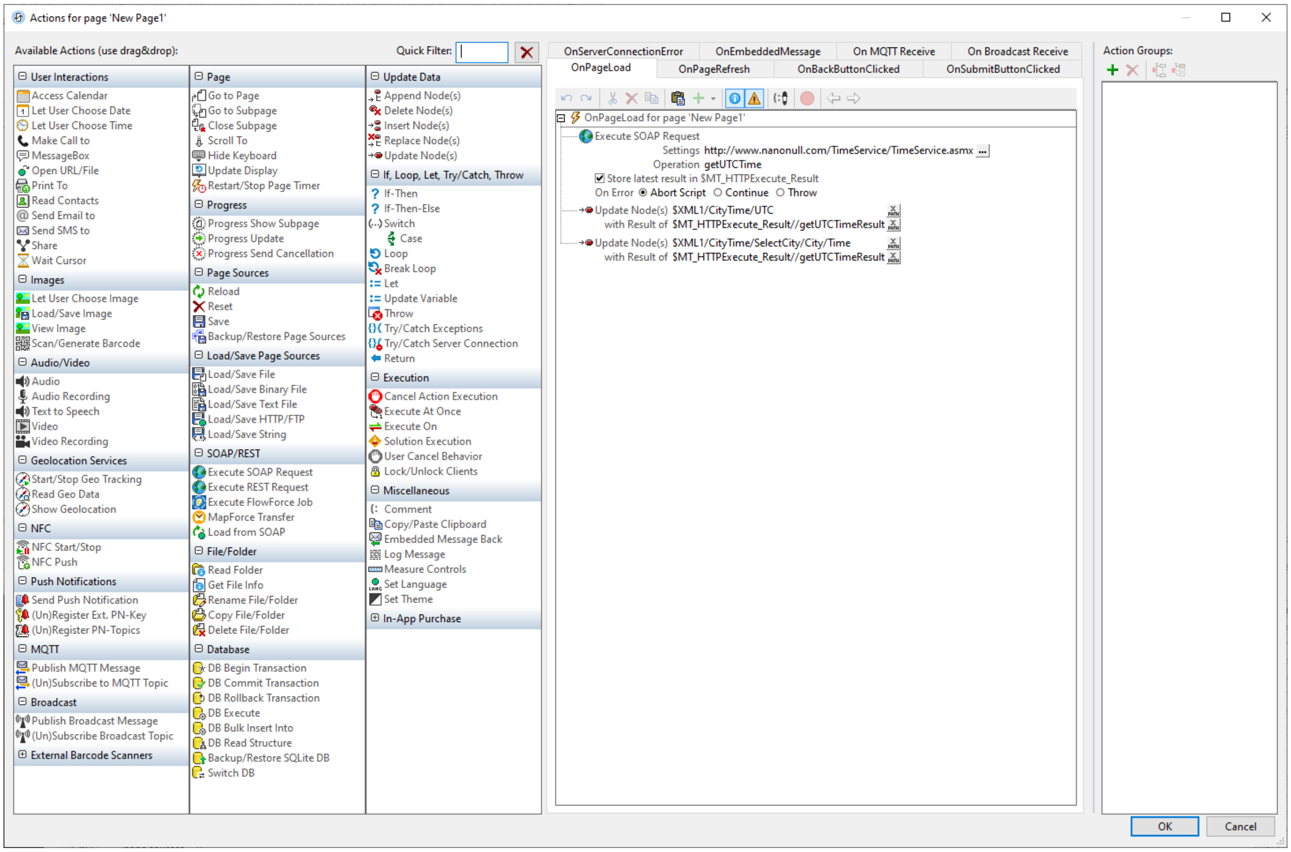
# 28.7 Page

The **Page** menu contains commands that apply to the currently active page of a project. You must click in the page to enables the commands of the menu. It contains the following commands:

- Page Actions 1631
- Actions Overview 1632
- Jump to Control 1633
- Show/Define Tab Order 1633
- List Text Size Auto-Fit Groups 1636

## 28.7.1 Page Actions

Displays the Actions dialog of the currently active page (*see screenshot below*). The left-hand pane of the dialog contains the available actions, organized by functionality. The right-hand pane contains tabs of available events for that page. The events that are available depend on the role of that page in the project workflow. For instance, a page that cannot be returned to by pressing the **Back** button will not have the `OnBackButtonClicked` event tab.

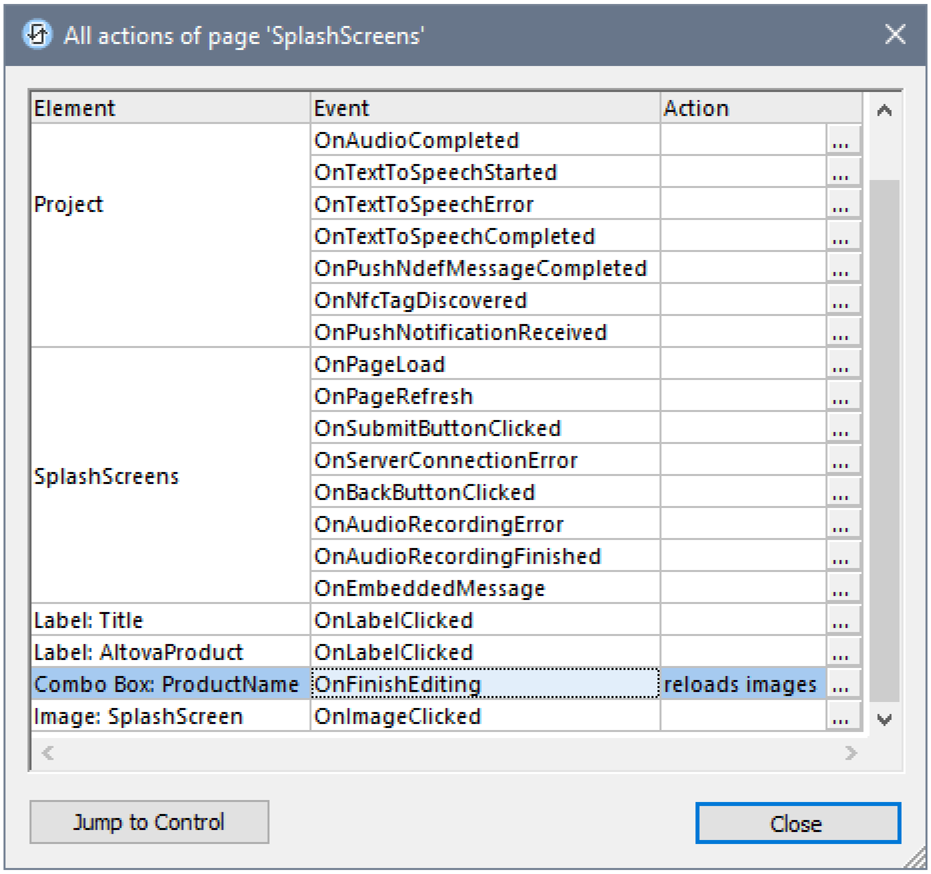


OnPageLoad for page 'New Page1'
Execute SOAP Request
Settings http://www.nanonull.com/TimeService/TimeService.asmx
Operation getUTCTime
☑ Store latest result in $MT_HTTPExecute_Result
On Error ● Abort Script ○ Continue ○ Throw
Update Node(s) $XML1/CityTime/UTC
with Result of $MT_HTTPExecute_Result//getUTCTimeResult
Update Node(s) $XML1/CityTime/SelectCity/City/Time
with Result of $MT_HTTPExecute_Result//getUTCTimeResult

To specify that a particular action (from among the available actions) is carried out when an available event occurs, drag the action from the left-hand pane into the event's tab in the right-hand pane. Specify additional properties of the action as required. (For more information about individual page actions, see the section, Page Actions[389].) Note that you can also add any Action Group that might be defined for the project[1631]. Click **OK** to finish.

## 28.7.2 Actions Overview

Displays the Actions Overview dialog of the currently active page (*see screenshot below*). The dialog shows Control Actions[665] and Page Actions[389], as well as actions you can set for the project[296]. Each control in the page design is listed, together with its event/s and corresponding action/s. If an action has been defined for an event, then that action is listed in the *Actions* column; otherwise no action is listed for the event. The screenshot below shows that only one action has been defined on a combo box event.



| Element | Event | Action |
|---|---|---|
| Project | OnAudioCompleted | ... |
| | OnTextToSpeechStarted | ... |
| | OnTextToSpeechError | ... |
| | OnTextToSpeechCompleted | ... |
| | OnPushNdefMessageCompleted | ... |
| | OnNfcTagDiscovered | ... |
| | OnPushNotificationReceived | ... |
| SplashScreens | OnPageLoad | ... |
| | OnPageRefresh | ... |
| | OnSubmitButtonClicked | ... |
| | OnServerConnectionError | ... |
| | OnBackButtonClicked | ... |
| | OnAudioRecordingError | ... |
| | OnAudioRecordingFinished | ... |
| | OnEmbeddedMessage | ... |
| Label: Title | OnLabelClicked | ... |
| Label: AltovaProduct | OnLabelClicked | ... |
| Combo Box: ProductName | OnFinishEditing | reloads images ... |
| Image: SplashScreen | OnImageClicked | ... |

The dialog is structured to show:

- Actions of project-wide events
- Page Actions[389]. In the screenshot above, for example, `SplashScreens` is the name of the project's one page. All the events of the page are listed. No action has been defined for any of these events.
- Control Actions[665]. Each control in the design is listed with its respective event.

This dialog not only provides you with an overview of all actions defined in the design, but also enables you to do the following:

- To create an action for any event or to edit an existing action: Click the **Edit** icon of that event (*see screenshot above*). This takes you to the corresponding Actions dialog of the control, page, or project.
- When a control is selected (not a page or the project), then the **Jump to Control** button is enabled. Clicking it takes you to that control in the design.

**Note:** The last selection in this dialog is remembered. As a result, the dialog is always reopened with the last selection highlighted.

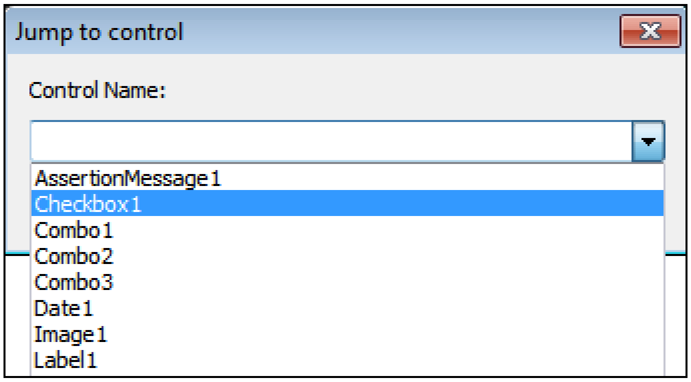
## 28.7.3 Jump to Control

*Shortcut*

**Ctrl+J**

*Description*

Displays the Jump to Control dialog (*screenshot below*). The dialog contains a combo box with a dropdown list that shows all the controls of the currently active page design. Page controls are listed alphabetically by the value of their `Name` properties.



Select a page control from the dropdown list or enter the control's name (auto-completion is available). Click **OK** to finish. The page control that was selected in the dialog box will now be selected in the design. If the control is associated with a data node, that node in the Page Sources Pane[270] will also be selected.

## 28.7.4 Show/Define Tab Order

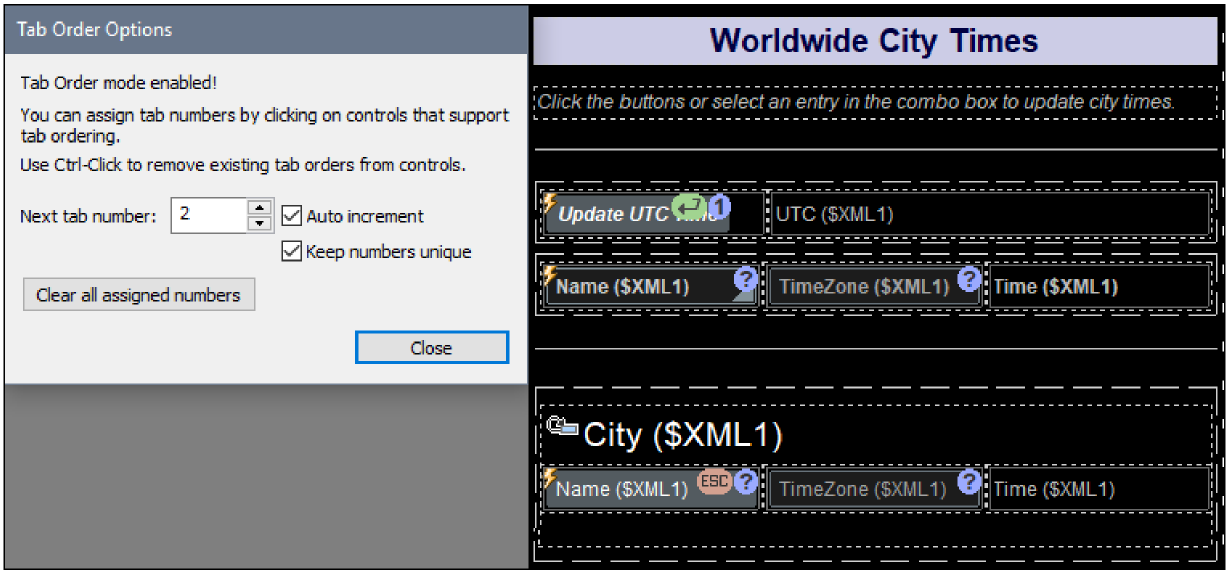
*Shortcut*

**Ctrl+T**

*Description*

The Tab Order feature enables you to define a tab order for the controls on the page. Once defined, every time the user of the device (Web and Windows clients only) taps the **Tab** key, the focus of the solution jumps to the next control in the tab order sequence.

When selected, this command does the following: (i) displays the Tab Order Options dialog (*screenshot below*), (ii) shows a blue number icon on all controls in the design that may be assigned a number in the tab order sequence (*see screenshot*), (iii) shows icons indicating **Enter** and **Escape** key shortcuts (green and pink icons, respectively) on those controls for which the control property `On Enter/Escape` has been set (*see screenshot*).



Tab Order Options

Tab Order mode enabled!

You can assign tab numbers by clicking on controls that support tab ordering.

Use Ctrl-Click to remove existing tab orders from controls.

Next tab number: 2 ☑ Auto increment

☑ Keep numbers unique

To set the tab order sequence, do the following:

1. Open the Tab Order Options dialog. The *Next tab number* field will display `1`. All the controls that may be assigned to the tab order sequence are indicated with a blue circle containing a question mark.
2. Click the control to which you wish to assign the number `1` in the tab order sequence. That control's icon will now contain the number `1`, and the number in the *Next tab number* field will increment to `2`.
3. Click the control that you want to be the second in the tab order sequence. That control's icon will now contain the number `2`, and the number in the *Next tab number* field will increment to `3`.
4. Continue to click controls in the sequence in which you want to define the entire tab number sequence.

Note the following points:

- The number in the *Next tab number* field will always be assigned to the next selection. This number can be manually changed or automatically incremented.
- The *Auto increment* option automatically increments the next number by one. If this option is not selected, then the *Next tab number* field does not increment and the number of the next control you select will depend on the value of the *Keep numbers unique* option.
- You can change the number of any individual control by setting its number in the *Next tab number* field and then clicking the control.

- The *Keep numbers unique* option ensures that the next selection is different from the previously assigned numbers.
- You can remove all the numbers assigned in the design by clicking **Clear all assigned numbers**.

**Note:** The Tab Order feature is available on Web and Windows clients only.

**Note:** The Tab Order feature can also be set for an individual control by selecting the control and setting its **`Tab Order`** property to the number in the sequence that you want for that control.

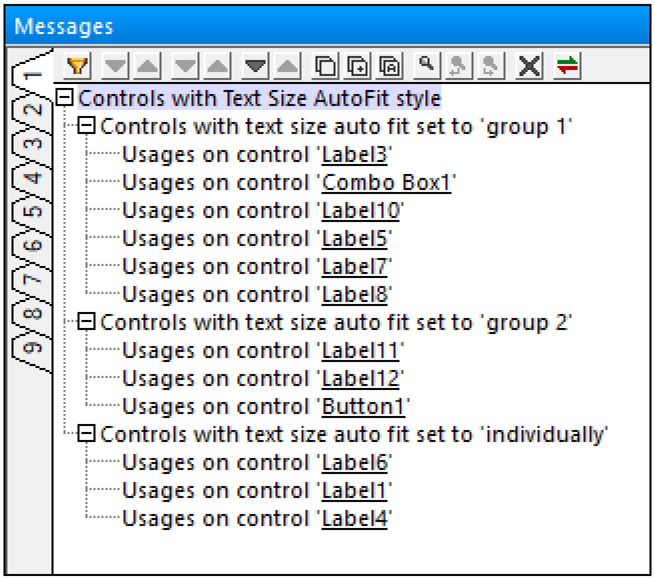
*Displaying the tab sequence, indicating controls assigned Enter/Escape shortcuts*

When the Tab Order Options dialog is opened, the following indicators in the design become visible:

- All controls that may be assigned to the tab order sequence are indicated with a blue circle containing a number giving its position in the sequence. If the control has not been assigned to the tab order sequence, the blue circle contains a question mark.
- Controls that have been assigned an **Enter** or **Escape** key shortcut are indicated with the respective shortcut indicators (*see screenshot above*). If more than one control has been assigned to the same (**Enter** or **Escape**) shortcut, then the shortcut will apply to the first visible and enabled control of the set.

## 28.7.5 List Text Size Auto-Fit Groups

Displays in the Listings Pane[281] a list of all the controls on the page for which the `Text Size Auto-Fit` property has been set, organized by auto-fit group (*see screenshot below*). For a description of the `Text Size Auto-Fit` property, see the description of any control[403] for which the property is available, for example, the Label[554] or Button[417] control.



Controls with Text Size AutoFit style
- Controls with text size auto fit set to 'group 1'
  - Usages on control 'Label3'
  - Usages on control 'Combo Box1'
  - Usages on control 'Label10'
  - Usages on control 'Label5'
  - Usages on control 'Label7'
  - Usages on control 'Label8'
- Controls with text size auto fit set to 'group 2'
  - Usages on control 'Label11'
  - Usages on control 'Label12'
  - Usages on control 'Button1'
- Controls with text size auto fit set to 'individually'
  - Usages on control 'Label6'
  - Usages on control 'Label1'
  - Usages on control 'Label4'

# 28.8 Table

The **Table** menu commands are available when a table cell—and, by extension, a table row or table column—is selected. It contains the following commands:

- Insert Table 1637
- Delete Table 1637
- Insert Row 1638
- Append Row 1638
- Delete Row 1638
- Insert Column 1638
- Append Column 1638
- Delete Column 1638
- Join Cell Left 1639
- Join Cell Right 1639
- Join Cell Above 1639
- Join Cell Below 1639
- Split Cell Horizontally 1639
- Split Cell Vertically 1639
- Show Add–Remove Buttons 1639
- Insert Table Header 1640
- Append Table Footer 1640
- Insert Table Leading Column 1640
- Append Table Trailing Column 1640
- Remove Table Header 1641
- Remove Table Footer 1641
- Remove Table Leading Column 1641
- Remove Table Trailing Column 1641
- Convert this Row to Repeating Row 1643
- Convert this Row to Static Row 1643
- Convert to Repeating Table 1643
- Convert to Non-Repeating Table 1643
- Convert this Column to Repeating Column 1643
- Convert this Column to Static Column 1643
- Border Settings 1644

For more information about tables, see the section Design Object/Features | Tables 1062.

## 28.8.1 Insert/Delete Table

The **Insert Table** and **Delete Table** commands are available when a table cell in any kind of table (static 1063, repeating 1065, or tables with dynamic rows 1069 and/or dynamic columns 1074) is selected.

▼ Insert Table

⊟ *Icon*

⊟ *Description*

The **Insert Table** command inserts a static table of dimensions 2x2 to the left of the selected cell.

▼ Delete Table

⊟ *Icon*

⊟ *Description*

The **Delete Table** command deletes the currently selected table.

## 28.8.2 Insert/Append/Delete Row/Column

The two **Insert**, two **Append**, and two **Delete** commands shown in the table below are available when a row or column of any kind of table (static 1063, repeating 1065, or tables with dynamic rows 1069 and/or dynamic columns 1074) is selected.

| | |
|---|---|
| **Insert Row** | **Insert Column** |
| **Append Row** | **Append Column** |
| **Delete Row** | **Delete Column** |

The **Insert** and **Append** commands enable you to respectively insert/append rows/columns relative to the currently selected cell. Note that rows and columns added in this way are static. This means, for example, that if one static row is added in the design, then it will result in one static row in the output. Of course, if the row is added within a repeating structure, then the static row will also repeat along with each iteration of the structure.

The **Delete Row** and **Delete Column** commands delete the currently selected row/column respectively.

These commands are also available as context menu commands 1087 when a table cell is selected.

## 28.8.3 Join/Split Cells

The four **Join** commands and two **Split** commands (shown in the table below) enable you to, respectively, join the currently selected cell with an adjacent cell or split the currently selected cell. They are available for the cells of all tables (static[1063], repeating[1065], or tables with dynamic rows[1069] and/or dynamic columns[1074]).

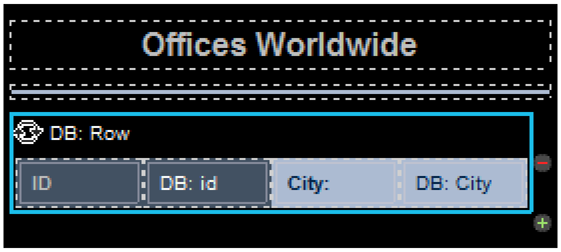
| | |
|---|---|
| **Join Cell Left** | **Split Cell Horizontally** |
| **Join Cell Right** | **Split Cell Vertically** |
| **Join Cell Above** | |
| **Join Cell Below** | |

*Joining cells*
You can join a cell with an adjacent cell, horizontally or vertically. Joining can be carried out multiple times as long as there are adjacent cells. Cell joining, in effect, creates cells that span horizontally across two or more columns or that span vertically across two or more rows. For additional information, see *Row/Column joining and spanning*[1083].

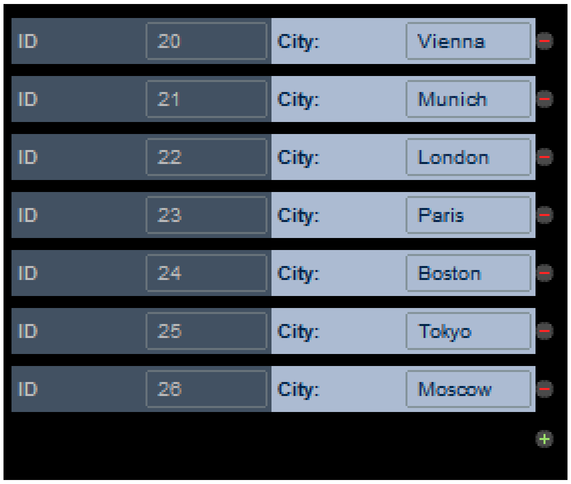
*Splitting cells*
You can split a cell horizontally or vertically into two cells. If a cell is split horizontally, then any content or formatting that was present in the original cell will be retained in the left-hand cell of the new pair; the right-hand cell will be empty. If the cell is split vertically, then the upper cell of the new cell pair will retain the content and formatting of the original cell; the lower cell will be empty.

## 28.8.4 Show Add–Remove Buttons

The **Show Add–Remove Buttons** command is available for repeating tables[1065] or the repeating rows of tables with dynamic rows[1069]. It creates Add–Remove buttons for the selected row. The screenshot below shows the design of a table with dynamic rows[1069] that has its Add–Remove buttons enabled (indicated by the two icons at the right bottom edge of the table).



While the screenshot above shows the table in the design, the screenshot below shows how such a table looks on the client device. A row can be deleted by tapping its **Delete** button (*see screenshot below*); this will cause the corresponding row data to be removed from the underlying page source. You can add a new row by tapping the **Add** button.



| | | | |
|---|---|---|---|
| ID | 20 | City: | Vienna |
| ID | 21 | City: | Munich |
| ID | 22 | City: | London |
| ID | 23 | City: | Paris |
| ID | 24 | City: | Boston |
| ID | 25 | City: | Tokyo |
| ID | 26 | City: | Moscow |

## 28.8.5 Add Header/Footer, Leading/Trailing Column

These four commands each add, respectively, a table's header, footer, leading column, or trailing column. They each apply to tables with either dynamic (repeating) rows[1069] or dynamic (repeating) columns[1074].

- Insert Table Header

  ⊟ *Icon*

  ⊟ *Description*

  Inserts an empty header row in tables with dynamic rows[1069] that do not yet have a header. If the table already has a header, then this command is disabled.

- Append Table Footer

  ⊟ *Icon*

  ⊟ *Description*

Appends an empty footer row in tables with dynamic rows[1069] that do not yet have a footer. If the table already has a footer, then this command is disabled.

### Insert Table Leading Column

*Icon*

*Description*

Inserts an empty leading column in tables with dynamic columns[1074] that do not yet have leading column. If the table already has a leading column, then this command is disabled. *(Note: If the leftmost column of a table acts as a vertical header-column, then it is referred to as the table's leading column.)*

### Append Table Trailing Column

*Icon*

*Description*

Appends an empty trailing column in tables with dynamic columns[1074] that do not yet have trailing column. If the table already has a trailing column, then this command is disabled. *(Note: If the rightmost column of a table acts as a vertical footer-column, then it is referred to as the table's trailing column.)*

## 28.8.6 Remove Header/Footer, Leading/Trailing Column

These four commands each remove, respectively, a table's header, footer, leading column, or trailing column. They each apply to tables with either dynamic (repeating) rows[1069] or dynamic (repeating) columns[1074].

### Remove Table Header

*Icon*

#### *Description*

Removes the header of tables with dynamic rows[1069]. If the table does not have a header, then this command is disabled.

### Remove Table Footer

#### *Icon*

#### *Description*

Removes the footer of tables with dynamic rows[1069]. If the table does not have a footer, then this command is disabled.

### Remove Table Leading Column

#### *Icon*

#### *Description*

Removes the leading column of tables with dynamic columns[1074]. If the table does not have a leading column, then this command is disabled. *(Note: If the leftmost column of a table acts as a vertical header-column, then it is referred to as the table's leading column.)*

### Remove Table Trailing Column

#### *Icon*

#### *Description*

Removes the trailing column of tables with dynamic columns[1074]. If the table does not have a trailing column, then this command is disabled. *(Note: If the rightmost column of a table acts as a vertical footer-column, then it is referred to as the table's trailing column.)*

## 28.8.7 Convert Row to Repeating/Static Row

▼ Convert This Row to Repeating Row

⊟ *Description*

If the selected row ***is not*** a dynamic (repeating) row[1069], then this command converts it to a repeating row[1069]. If the selected row is a repeating row[1069], then this command is disabled.

▼ Convert This Row to Static Row

⊟ *Description*

If the selected row ***is*** a dynamic (repeating) row[1069], then this command converts it to a static row[1063]. If the selected row is not a repeating row[1069], then this command is disabled.

## 28.8.8 Convert to Repeating/Non-Repeating Table

▼ Convert to Repeating Table

⊟ *Description*

If the selected table ***is not*** a repeating table[1065], then this command converts it to a repeating table[1065]. If the selected table is a repeating table[1065], then this command is disabled.

▼ Convert to Non-Repeating Table

⊟ *Description*

If the selected table ***is*** a repeating table[1065], then this command converts it to a non-repeating table[1063]. If the selected table is not a repeating table[1065], then this command is disabled.

## 28.8.9 Convert Column to Repeating/Static Column

▼ Convert this Column to Repeating Column

⊟ *Description*

If the selected column ***is not*** a dynamic (repeating) column[1074], then this command converts the column to a repeating column[1074]. If the selected column is a repeating column[1074], then this command is disabled.

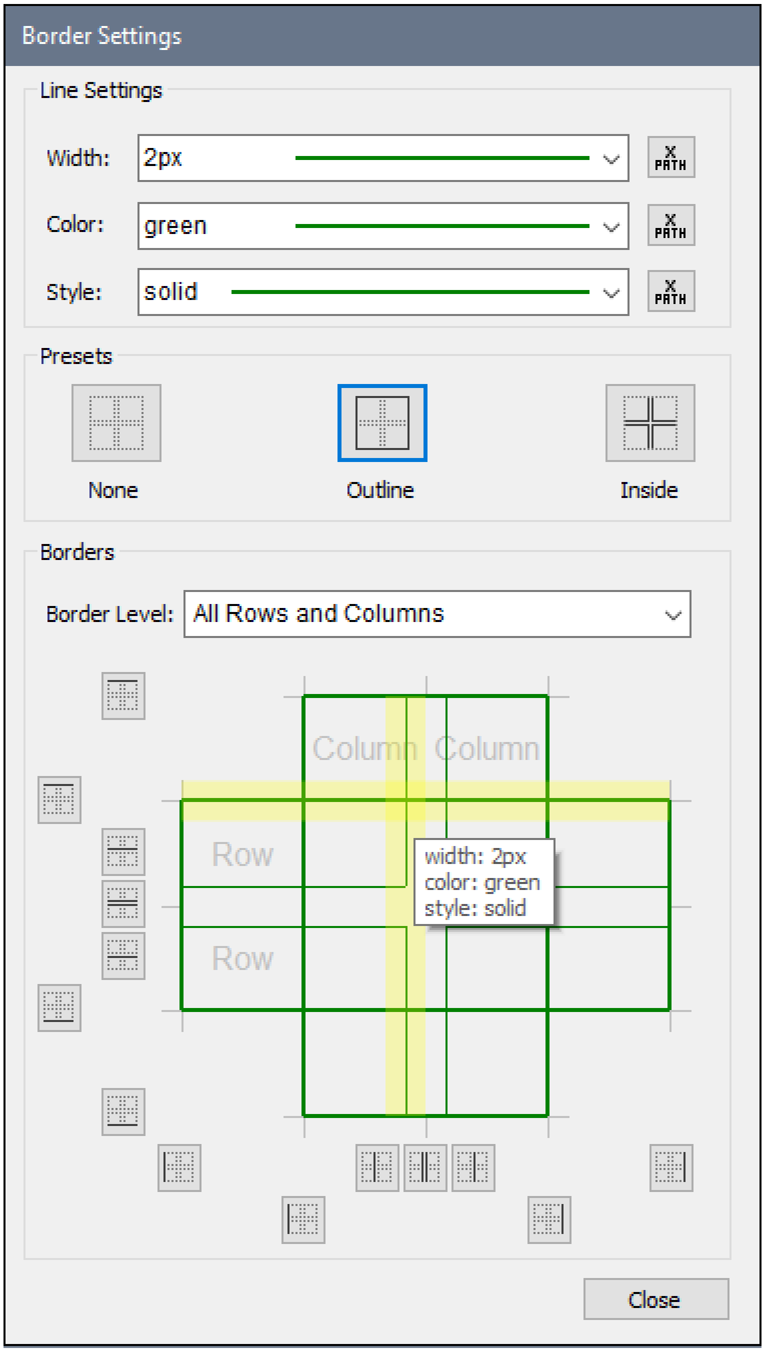
▼ Convert this Column to Static Column

⊟ *Description*

If the selected column ***is*** a dynamic (repeating) column[1074], then this command converts the column to a static column[1063]. If the selected column is not a repeating column[1074], then this command is disabled.

## 28.8.10 Border Settings

The **Border Settings** command opens the Border Settings dialog (*screenshot below*), in which you can set border properties for the selected table item/s and related table items (for example, the table items that are related to a cell are the rows, columns, row groups, and table that contain the cell).



Border Settings

Line Settings

Width: 2px

Color: green

Style: solid

Presets

Borders

Border Level: All Rows and Columns

width: 2px
color: green
style: solid

The mechanism to set border styles works as follows:

1. In the **design**, select the table item or multiple table items[1646] that you want to style: cell/s, row/s, column/s, row-group/s, table.
2. In the Line Settings pane[1646], define the three border properties: width, color, and style that you want to apply.
3. The Border Level combo box[1647] of the Borders pane displays table items according to the table item you selected in Step 1. In the combo box, choose the selected table item or one of the options for related table items. The style will be applied to the table item/s that you select in the combo box.

4. Apply the border styles 1647 to one or more individual borders (top, right, bottom, left) of the currently selected table items. Use the Presets pane and/or Borders pane to apply styles. Note that the buttons in these two panes are toggles; they apply/remove styles to the respective border.
5. To set new border styles (for non-styled borders or already styled borders), repeat Steps 1 to 4.
6. Click **Close** to finish.

These steps are explained in more detail below.

## Selection of table item/s

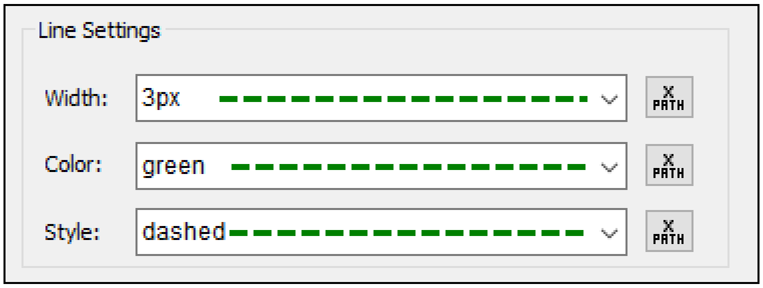
Select table items in the design as follows:

- *Cell:* Click inside the cell.
- *Multiple cells:* Click the first cell; keep the **Ctrl** key pressed and click the other cells that you want to make part of the selection.
- *Row, column, row-group:* Click inside a cell of the respective item. Alternatively, click the border of the item.
- *Multiple rows, columns, row-groups:* Select the first item; keep the **Ctrl** key pressed and click the other items.
- *Table:* Click inside a cell of the table. Alternatively, click the border of the table.

After the table item is selected in the design, go to the Border Level combo box 1647 and choose the context-relevant items to which the styles should apply.

## Line settings

Borders have three properties: width, color, and style. You can select the value of each property from its combo box. Alternatively, you can enter an XPath expression to select the value of a property.

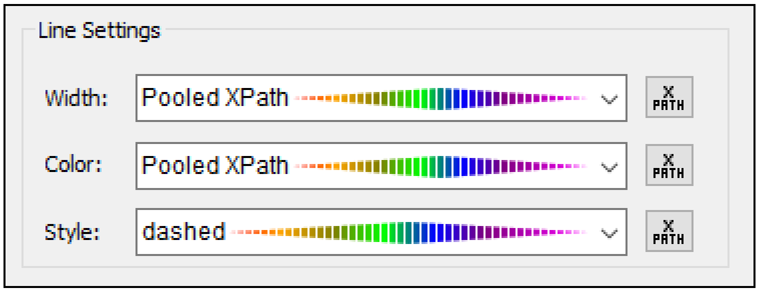


Line Settings

Width: 3px

Color: green

Style: dashed



Line Settings

Width: Pooled XPath

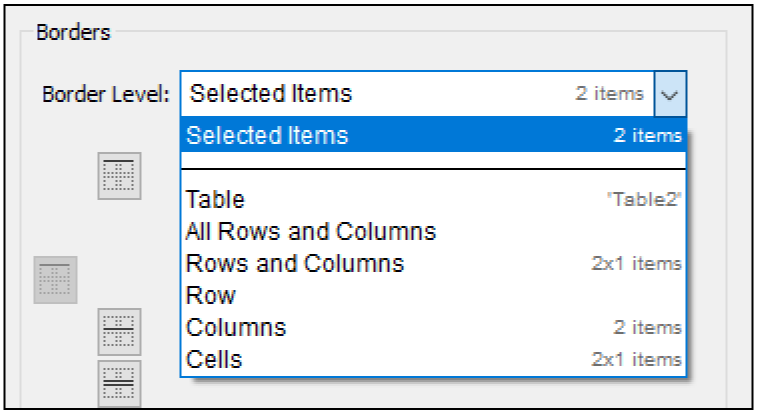
Color: Pooled XPath

Style: dashed

The resulting border style is displayed as a preview. In the screenshot above left, for example, the preview line is green, dashed, and has a width of 3px. If a property is specified as an XPath expression, then this is indicated by *Pooled XPath*, and, since the property value is not known till run time, the property is indeterminate in the preview. (An example of such a preview is the screenshot above right, where the style property is dashed (and shown as such), whereas the width and color are indeterminate.) At run time, the evaluated values will be applied to the respective border properties.

## Border level

In the Borders pane, the Border Level combo box enables you to select items that are related to the table items currently selected in the design (*see screenshot below*).



The combo box items from which you can choose are contextually related to the table item/s currently selected in the design. To the right of each item in the combo box, additional information provides detail about the table structure. For example, we can see that, in the screenshot above, the items that are selected in the design are two adjacent cells in a row of Table2. The information in the last line, **`2x1 items`**, indicates that the two cells are each in a separate column but in a single row. If the two cells were in one column and in two rows, the information would read **`1x2 items`**. (The first number is the number of columns, the second is the number of rows.) Note that the *Rows and Columns* item also shows information in the `column x row` format.

In the combo box, select the table item/s to which you want to apply the style[1647] you have defined in the Line Settings pane[1646]. If, for example, you select *Row*, then the style will be applied to the border/s of the currently selected row. If you select *Cells*, then the style will be applied to the border/s of the currently selected cells, individually.

## Applying styles to (individual) borders of selected table items

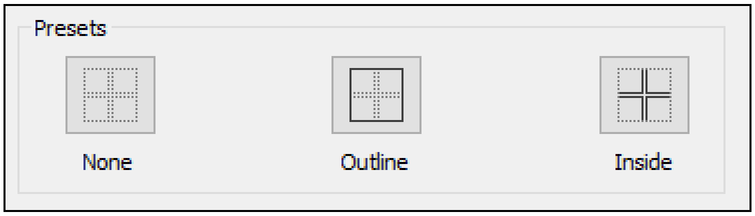
The style that is currently defined in the Line Settings pane[1646] can be applied in two ways to borders of the table items selected in the Border Level combo box[1647]:

- By selecting one of the presets in the Presets pane
- By selecting a combination of individual borders in the Borders pane

The options provided by the two panes are described below.

**Note:** The application options in both panes are toggles. This means that, if the **current style** (exactly as defined in the Line Settings pane[1646]) has not been applied, then clicking an option applies the current style; if the **current style** has been applied (exactly as defined in the Line Settings pane[1646]), then clicking an option removes the current style.

*Presets pane*
Three preset options (*see screenshot below*) are available, and are described below. Note that only applicable options are enabled.

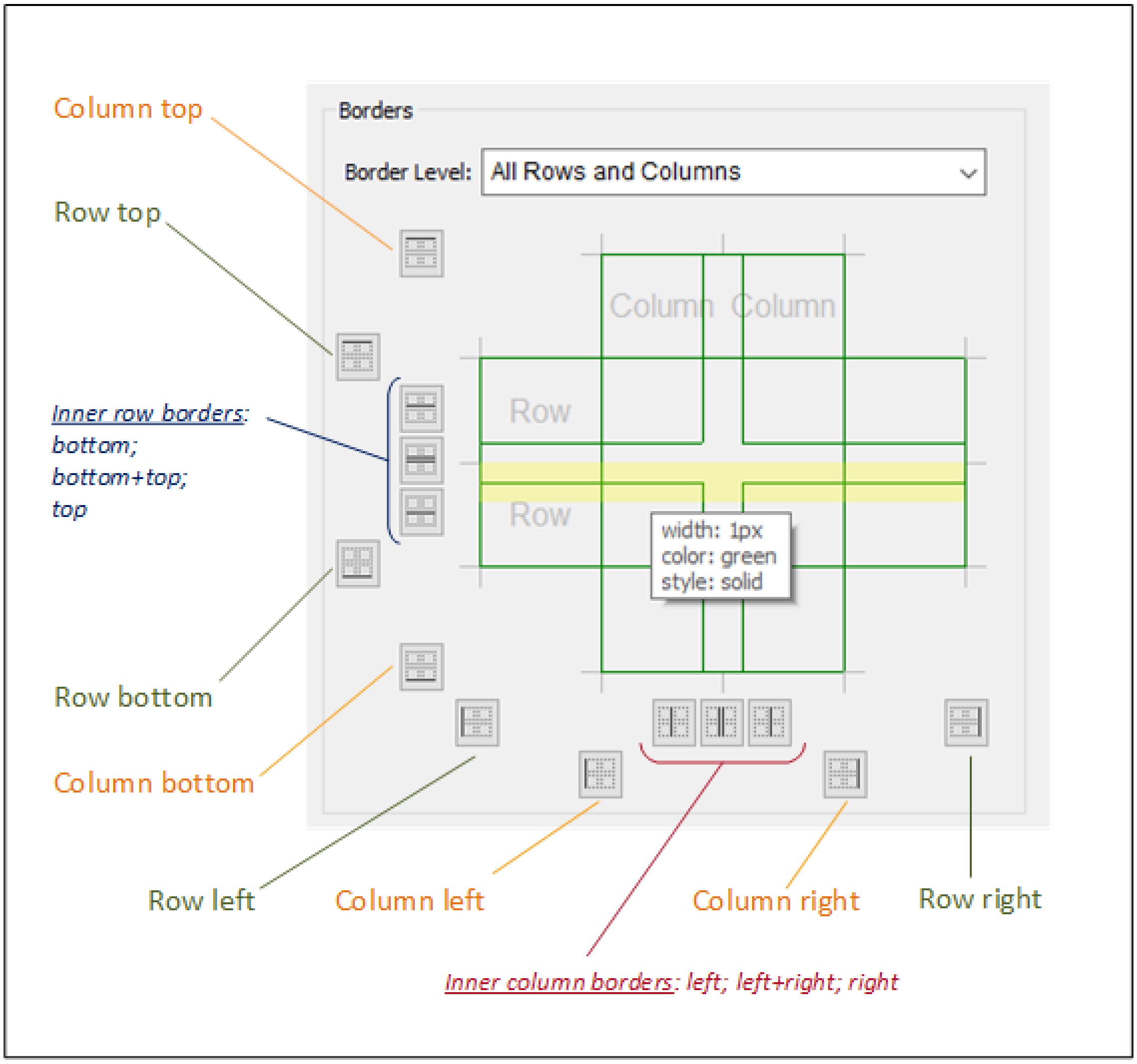


- *None:* This preset option is always enabled. It removes all border styles from the selected table item. If you want to remove border styles from any table item, this option is the best way to do it.
- *Outline:* This preset option is always enabled. It applies the current style[1646] to the outer border of the selected item (on all sides). If the selected item is a table, then the table border gets the style. If two cells are selected, then both cells, individually, get a border in the current style[1646].
- *Inside:* This option is enabled only when at least two adjacent cells/rows/columns are selected; it is not applicable to tables. For example: (i) if applied to two columns, then the current style[1646] is applied only to the inside vertical borders of all cells in the column; (ii) if applied to two horizontally adjacent cells, then the inside vertical borders of both cells get the style; (iii) if applied to two vertically adjacent cells, then the inside horizontal borders of both cells get the style.

*Borders pane*
Options in the Borders pane enable you to apply the current style[1646] to individual borders of a table item (that is, top, right, bottom, left borders separately). Each individual border has its own icon. Click an icon to apply the current style[1646] to that border.

Note the following points:

- For each table item selected in the Border Level combo box[1647], a relevant set of icons is enabled. For example, if a single cell is selected, then only four icons will be enabled (corresponding to the top, right, bottom, and left borders). If, on the other hand, however, *All Rows and Columns* is selected, then all 14 icons in the pane are enabled (*see screenshot below*). Note that the inner borders can be styled separately (for example: bottom or top inner row borders) or together (bottom and top inner row borders). Also note that, strictly speaking, columns themselves are not given a top or bottom border; it is the topmost cell in the column that is given a top border and the bottom-most cell of the column that is given a bottom border. This works analogously for the left and right borders of rows.



- Cells of a column or row will inherit the style of that column or row, respectively.
- Individual table borders can also be applied by moving the mouse, in the Borders pane, over the location of the border you want to style. When the border becomes highlighted in yellow *(see screenshot above)*, click to apply the current style[1646]. If you move the mouse over the intersection of two borders, then both borders become highlighted (*see the first screenshot at the top of this page*), and you can apply the current style to both borders at once.
- After a style has been applied to a border and you move the mouse over the border, in addition to the border becoming highlighted in yellow, the border's properties are displayed in a popup *(see screenshot above)*.
- When you click an icon or a highlighted border, the current style[1646] is applied if that border does not have the same style as the current style. If the border already has the current style[1646] and you click the border or its icon, then the current style is removed from that border.

# 28.9 View

The **View** menu contains the following commands:

- Status Bar 1650
- Pages 1650
- Files 1650
- Controls 1650
- Modules 1650
- Page Sources 1650
- Overview 1650
- Styles & Properties 1650
- Messages 1650
- Listings 1650
- Find & Replace 1650
- All On/Off 1650
- Back 1651
- Forward 1651
- Zoom 1651
- Zoom In 1651
- Zoom Out 1651
- Zoom Reset to 100% 1651
- Zoom to Selection 1651
- Fit to Window 1651

## 28.9.1 Status Bar and Panes

The display of the Status Bar and various panes can be toggled on/off by clicking their commands in the View Menu:

- Status Bar
- Pages Pane 257
- Files Pane 259
- Modules Pane 263
- Controls Pane 266
- Page Sources Pane 270
- Overview Pane 272
- Styles & Properties Pane 274
- Messages Pane 278
- Listings Pane 281
- Find & Replace Pane 283

The **All On/Off** command toggles all the panes and the status bar together between being displayed and being hidden.

### 28.9.2 Back, Forward

- Back

  ⊟ *Icon*

  ⇦

  ⊟ *Shortcut*

  **Alt + Arrow-Left**

  ⊟ *Description*

  Goes to the page or subpage that was viewed previously.

- Forward

  ⊟ *Icon*

  ⇨

  ⊟ *Shortcut*

  **Alt + Arrow-Right**

  ⊟ *Description*

  Goes to the page or subpage that was viewed before going back to the current page.

### 28.9.3 Zoom Levels

The zoom commands taken together provide considerable flexibility in changing the magnification level of the page design (between 10% and 100%) and the workflow diagram (between 10% and 200%). The zoom commands are enabled in Page Design View[253].

- Zoom

  ⊟ *Description*

  Opens the Zoom dialog, which contains a slide rule that enables you to change the magnification from 10% to 100% in Page Design View[253].

## Zoom In

### *Icon*

### *Description*

Magnifies the diagram by 10 percent points (`100`, `110`, `120...`) each time the command is executed. In Page Design View[253], there might be an upper limit due to template size constraints.

## Zoom Out

### *Icon*

### *Description*

Reduces the diagram by 10 percent points (`100`, `90`, `80...`) each time the command is executed.

## Zoom Reset to 100%

### *Description*

Resets the zoom factor to 100%. This is a quick way to regain the original size.

# 28.10 Tools

The **Tools** menu contains the following commands:

- Global Resources 1653
- Active Configuration 1654
- User-Defined Tools 1655
- Customize 1655
- Restore Toolbars and Windows 1663
- Options 1663

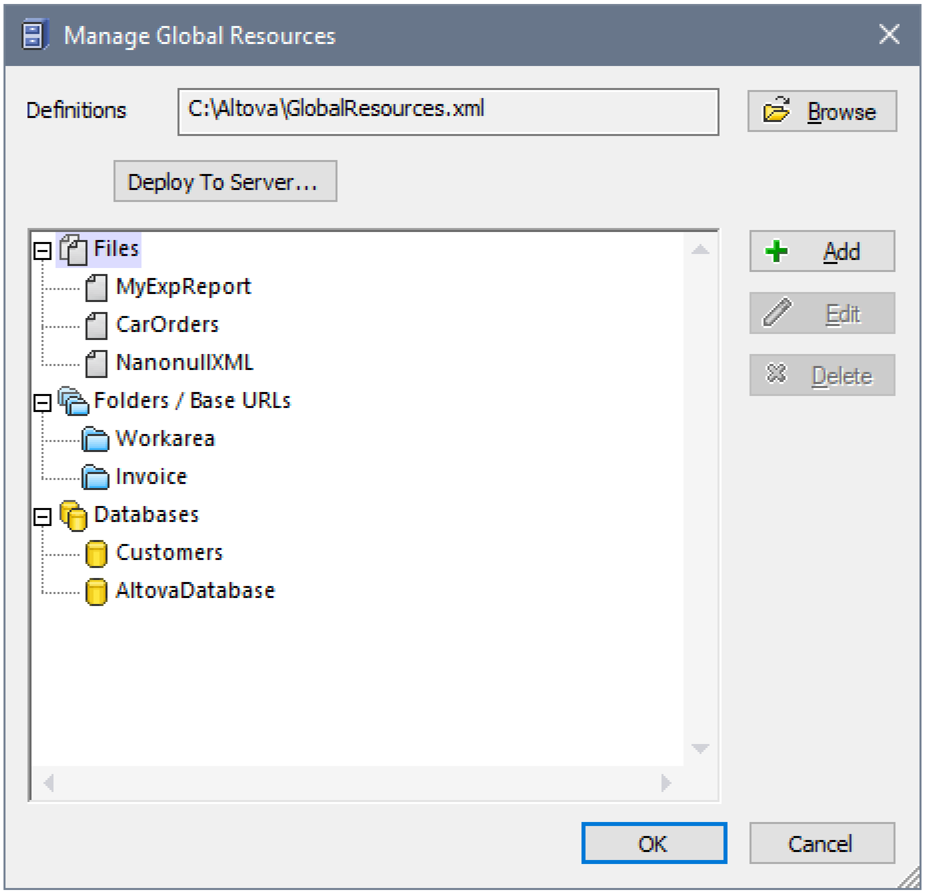
## 28.10.1 Global Resources

⊟ *Icon*

⊟ *Description*

Displays the Global Resources dialog (*screenshot below*), in which you can:

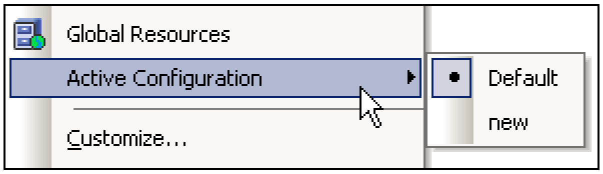
- Specify the Global Resources XML File to use for global resources (*Definitions file*).
- Add file, folder, and database global resources (or aliases)
- Specify various configurations for each global resource (alias). Each configuration maps to a specific resource. (Edit a global resource to do this.)
- **Deploy to Server** enables you to deploy a global resource to a MobileTogether Server.



How to define global resources is described in detail in the section, Defining Global Resources[1334].

## 28.10.2 Active Configuration

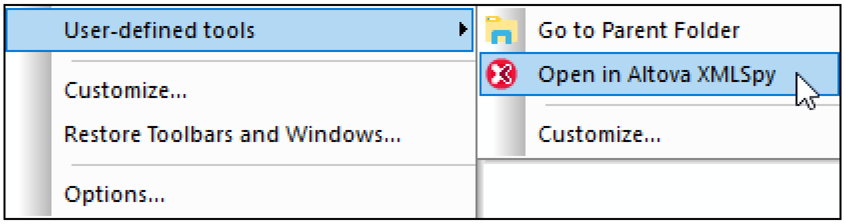
Placing the mouse over the command rolls out a submenu containing all the configurations defined in the currently active Global Resources XML File[1335].



The currently active configuration is indicated with a bullet. In the screenshot above the currently active configuration is `Default`. To change the active configuration, select the configuration you wish to make active.

## 28.10.3 User-Defined Tools

Placing the cursor over the **User-defined Tools** command rolls out a sub-menu containing custom-made commands that use external applications. In the screenshot below, two custom commands: (i) to open the parent folder in Windows Explorer, and (ii) to open the active file in Altova XMLSpy. You can create these commands in the Tools tab of the Customize dialog 1658. Clicking one of these custom commands executes the action associated with this command.



Below the listing of custom commands that you have created, is the **Customize** command (*see screenshot*). This opens the Tools tab of the Customize dialog 1658, enabling you to go directly to the dialog in which you can edit an existing custom command or create a new custom command.

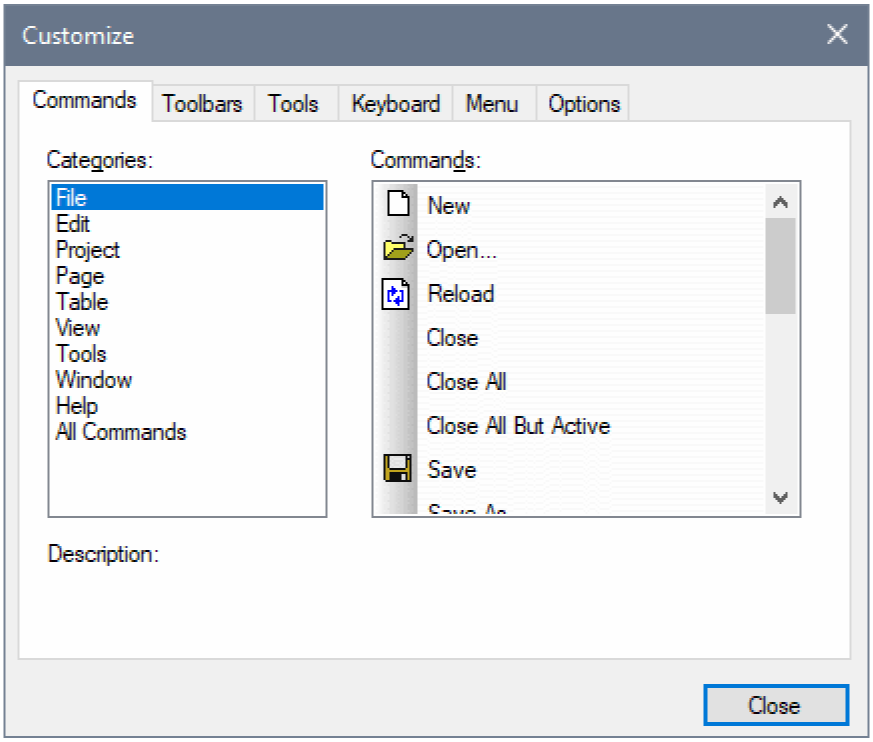
## 28.10.4 Customize

The customize command lets you customize MobileTogether Designer to suit your personal needs.

- Commands 1655
- Toolbars 1657
- Tools 1658
- Keyboard 1660
- Menu 1661
- Options 1663

### 28.10.4.1 Commands

The Commands tab enables you to add application commands to menus and toolbars according to your preference. Note that you cannot create new application commands or menus yourself.



To add a command to a toolbar or menu, do this:

1. Select the **All Commands** category in the *Categories* list box. The available commands appear in the *Commands* list box.
2. Click on a command in the *Commands* list box and drag it to an existing menu or toolbar. An **I**-beam appears when you place the cursor over a valid position to drop the command.
3. Release the mouse button at the position you want to insert the command.

   Note the following:

   - When you drag a command, a small button appears at the tip of mouse pointer: This indicates that the command is currently being dragged.
   - An "**x**" below the pointer indicates that the command cannot be dropped at the current cursor position.
   - If the cursor is moved to a position at which the command can be dropped (a toolbar or menu), the "**x**" disappears and an **I**-beam indicates the valid position.
   - Commands can be placed in menus or toolbars. If you have created you own toolbar 1657, you can use this customization mechanism to populate the toolbar.
   - Moving the cursor over a closed menu, opens that menu, allowing you to insert the command anywhere in that menu.

To add a command to a context menu, do this:

1. In the Customize dialog, click the **Menu** 1661 tab.
2. In the Context Menu pane, select a context menu from the combo box. The selected context menu appears.
3. In the Customize dialog, switch back to the Commands tab.

4. Drag the command you wish to create from the *Commands* list box and drop it onto the desired location in the context menu.

To delete a command from a menu, context menu, or toolbar, or to delete an entire menu, do this.

1. With the Customize dialog open (and any tab selected), right-click a menu or a menu command.
2. Select **Delete** from the context menu that appears. Alternatively, drag the menu or menu command till an "**x**" icon appears below the mouse pointer, and then drop the menu or menu command.

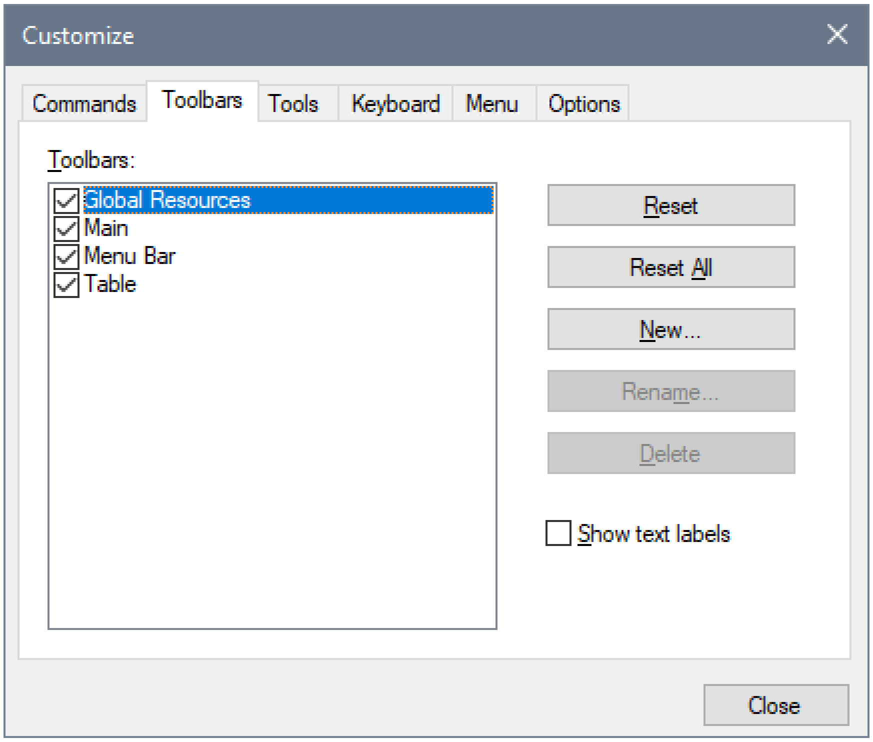
To reinstate deleted menu commands, use the mechanisms described in this section. To reinstate a deleted menu, go to **Tools | Customize | Menu**, and click the **Reset** button in the *Application Frame Menus* pane. Alternatively, go to **Tools | Customize | Toolbars**, select Menu Bar, and click the **Reset** button.

## 28.10.4.2 Toolbars

Application toolbars contain icons for the most frequently used menu commands. Information about each icon is displayed in a tooltip and in the Status Bar when the cursor is placed over the icon. You can drag a toolbar to any location on the screen, where it will appear as a floating window.

The Toolbars tab enables you to do the following:

- Activate or deactivate specific toolbars (that is, to decide which ones to display in the interface)
- Set what icons are displayed in each toolbar
- Create your own specialized toolbars



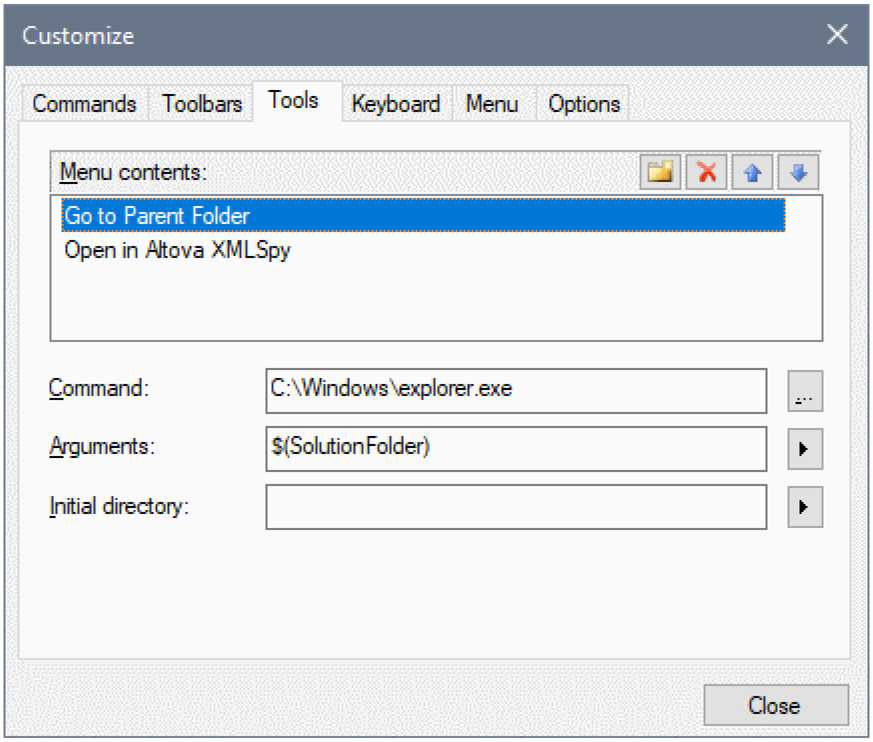
The following functionality is available:

- *To activate or deactivate a toolbar:* Click its check box in the *Toolbars* list box.
- *To add a new toolbar*: Click the **New** button and give the toolbar a name in the Toolbar Name dialog that pops up. From the Commands[1655] tab drag commands into the new toolbar.
- *To change the name of an added toolbar:* Select the added toolbar in the Toolbars pane, click the **Rename** button, and edit the name in the Toolbar Name dialog that pops up.
- *To reset the Menu bar*: Select the *Menu Bar* item in the Toolbars pane, and then click **Reset**. This resets the Menu bar to the state it was in when the application was installed.
- *To reset all toolbar and menu commands:* Click the **Reset All** button. This resets all toolbars and menus to the states they were in when the application was installed.
- *To delete a toolbar:* Select the toolbar you wish to delete in the Toolbars pane and click **Delete**.
- *To show text labels of commands in a particular toolbar*: Select that toolbar and click the *Show Text Labels* check box. Note that text labels have to be activated for each toolbar separately.

**Note:** To add a command to a toolbar, drag the command you want from the *Commands* list box in the Commands[1655] tab to the toolbar. To delete a command from a toolbar, open the Customize dialog, and with any tab selected, drag the command out of the toolbar (see Commands[1655] for more details).

### 28.10.4.3 Tools

The **Tools** tab allows you to set up commands to use external applications from within MobileTogether Designer. These commands will be added to the **Tools | User-defined Tools**[1655] menu. For example, the active file in MobileTogether Designer can be opened in an external application, such as XMLSpy, by clicking a command in the **Tools | User-defined Tools** menu that you created in this (the Tools) tab.



Menu contents:
Go to Parent Folder
Open in Altova XMLSpy

Command: C:\Windows\explorer.exe
Arguments: $(SolutionFolder)
Initial directory:

To set up a command to use an external application, do the following:

1. In the *Menu Contents* pane (*see screenshot above*), click the **New** icon in the title bar of the pane and, in the item line that is created, enter the name of the menu command you want. In the screenshot above, we have entered a menu command, **Go to Parent Folder**. We plan to use this command to open the parent folder of the active document in Windows Explorer. More commands can be added to the command list by clicking the **New** icon. In the screenshot above, for example, a command was created to open the active document in Altova's XMLSpy program. A command can be moved up or down the list relative to other commands by using the **Move Item Up** and **Move Item Down** icons. To delete a command, select it and click the **Delete** icon.
2. To associate an external application with a command, select the command in the *Menu Contents* pane. Then, in the *Command* field, enter the path to, or browse for, the executable file of the external application. In the screenshot above, the path to the Windows Explorer executable file has been entered in the *Command* field.
3. The argument to be passed to the external application is selected in the *Arguments* field (*see screenshot above*). The available arguments are displayed when you click the flyout button of the *Arguments* field and are described in the list below. When you select an argument, a code string for it is entered in the *Arguments* field. For example, in the screenshot above, the argument passed to Windows Explorer as the folder to open is the folder in which the active document—which must be a solution design—is located
4. If you wish to specify a current working directory (optional), enter it in the *Initial Directory* field.
5. Click **Close** to finish.

The user-defined command/s you created will appear in the **Tools | User-defined Tools**[1655] menu.

When you click a user-defined command (in the **Tools | User-defined Tools**[1655] menu) that you created, the action you associated with the command will be executed. The command example shown in the screenshot above does the following: It opens, in Windows Explorer, the folder in which the active solution is located. The Open in Altova XMLSpy command opens the active document in XMLSpy. The argument passed to XMLSpy is not the path to the parent folder, but the path to the active document.

## Arguments

The *Arguments* field specifies the argument to be passed to the external application command. The following arguments are available.

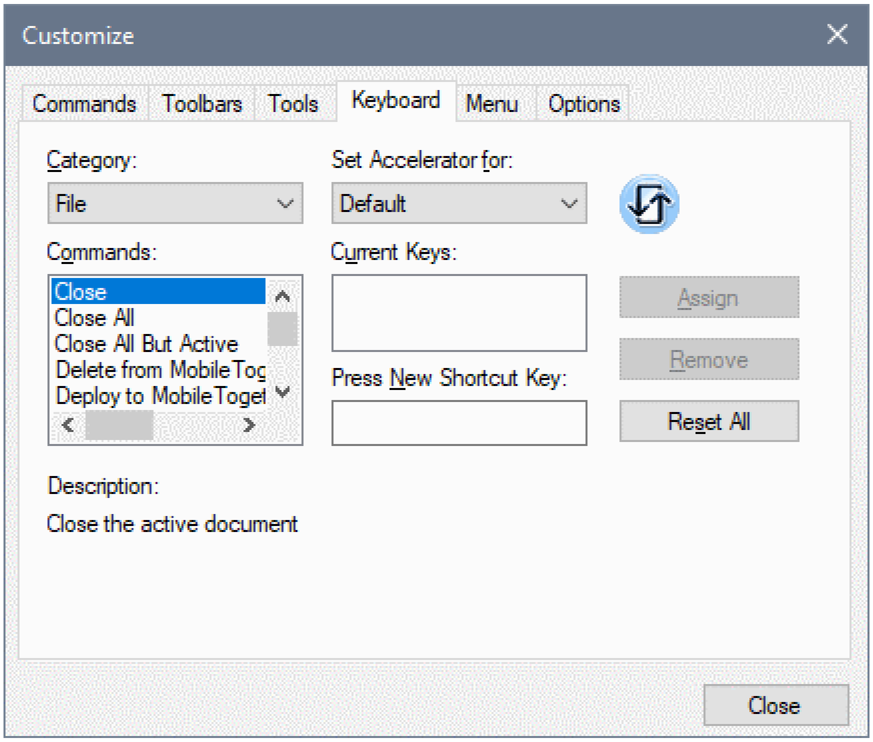
- *Solution File Name:* The name of the active solution design file.
- *Solution File Path:* The path to the active solution design file, including the name of the active file.
- *Solution Folder:* The parent folder of the active solution design file.
- *Temporary Folder:* The path to the Windows system folder that is used to store temporary files. On Windows.

## Initial directory

The *Initial Directory* entry is optional. It sets the initial directory for the command that is created.

## 28.10.4.4 Keyboard

The Keyboard tab enables you to create new keyboard shortcuts for any command.



Category: File

Set Accelerator for: Default

Description:

Close the active document

### Assign a shortcut

To assign a shortcut to a command, do the following.

1. Select the *All Commands* category in the *Category* combo box. Alternatively, select the menu you want to customize.
2. In the *Commands* list box, select the command to which you wish to assign a new shortcut or select the command the shortcut of which you wish to change.
3. Click in the *Press New Shortcut Key* text box, and press the shortcut you wish to assign to that command. The shortcut appears in the *Press New Shortcut Key* text box. If the shortcut has not yet been assigned to any command, the **Assign** button is enabled. If the shortcut has already been assigned to a command, then that command is displayed below the text box and the **Assign** button is disabled. (To clear the *Press New Shortcut Key* text box, press any of the control keys, **Ctrl**, **Alt** or **Shift**).
4. Click the **Assign** button to assign the shortcut. The shortcut now appears in the *Current Keys* list box. You can assign multiple shortcuts to a single command.
5. Click the **Close** button to confirm.

### Delete a shortcut

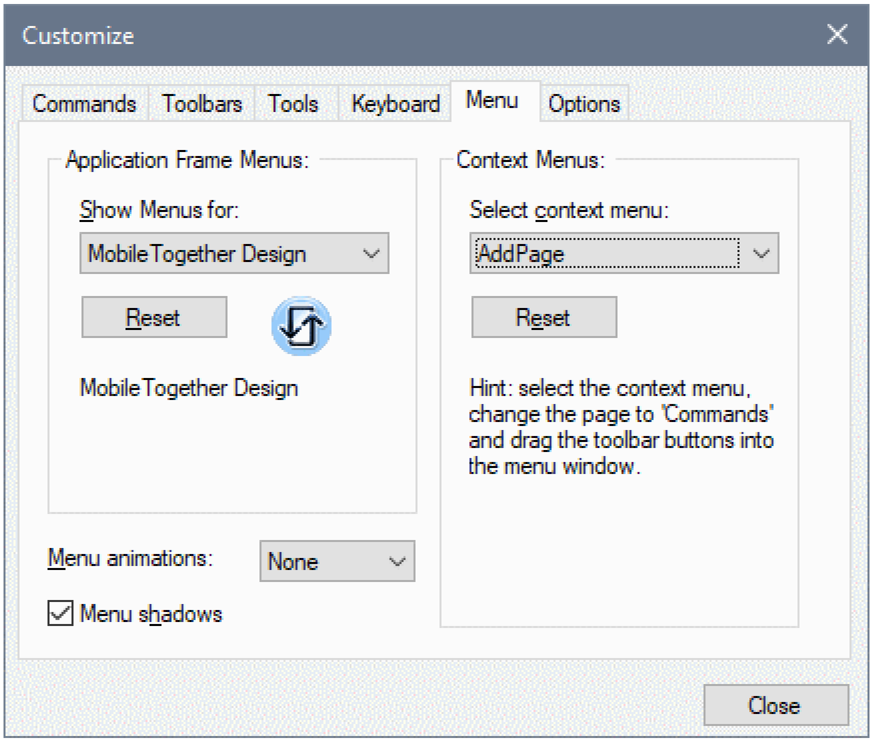
A shortcut cannot be assigned to multiple commands. If you wish to delete a shortcut, click it in the Current Keys list box and then click the **Remove** button. Click **Close**.

### Set accelerator for

Currently no function is available.

## 28.10.4.5 Menu

The Menu tab allows you to customize the two main menu bars (default and application menu bars) as well as context menus.



Application Frame Menus:

Show Menus for: MobileTogether Design

MobileTogether Design

Context Menus:

Select context menu: AddPage

Hint: select the context menu, change the page to 'Commands' and drag the toolbar buttons into the menu window.

Menu animations: None

☑ Menu shadows

### Customizing the default menu bar and application menu bar

The default menu bar is the menu bar that is displayed when no document is open in the main window. The application menu bar is the menu bar that is displayed when one or more documents are open in the main window. Each menu bar can be customized separately, and customization changes made to one do not affect the other. To customize a menu bar, select it in the *Show Menus For* combo box (*see screenshot above*). Then switch to the Commands tab of the Customize dialog 1655 and drag commands from the Commands list box to the menu bar or into any of the menus.

## Deleting commands from menus and resetting the menu bars

To delete an entire menu or a command inside a menu, select that menu or menu command, and then either (i) right-click and select **Delete**, or (ii) drag away from the menu bar or menu, respectively. You can reset each of these two menu bars (default and application menu bars) to its original installation state by selecting the menu in the *Show Menus For* combo box and then clicking the **Reset** button below the combo box.

## Customizing the application's context menus

Context menus are the menus that appear when you right-click certain objects in the application's interface. Each of these context menus can be customized by doing the following:

1. Select the context menu you want in the *Select Context Menu* combo box. This pops up the context menu.
2. Switching to the Commands tab of the Customize dialog 1655.
3. Drag a command from the *Commands* list box into the context menu.
4. If you wish to delete a command from the context menu, right-click that command in the context menu, and click **Delete**. Alternatively, you can drag the command you want to delete out of the context menu.

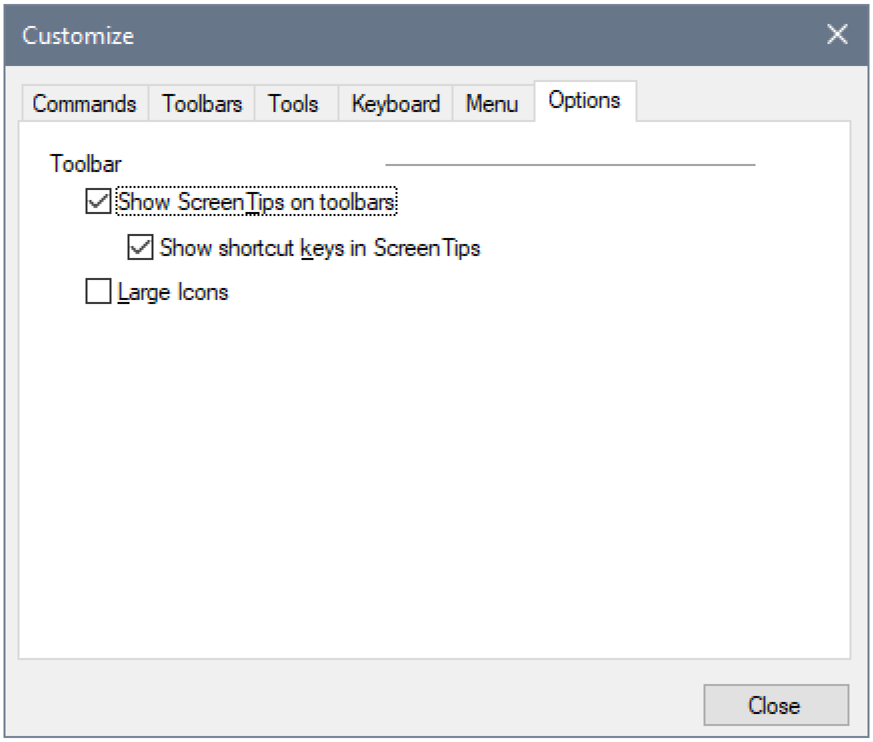
You can reset any context menu to its original installation state by selecting it in the *Select Context Menu* combo box and then clicking the **Reset** button below the combo box.

## Menu shadows

Select the *Menu shadows* check box to give all menus shadows.

### 28.10.4.6 Options

The Options tab allows you to define general environment settings.



Toolbar

☑ Show ScreenTips on toolbars

☑ Show shortcut keys in ScreenTips

□ Large Icons

Select the check boxes to toggle on the following options:

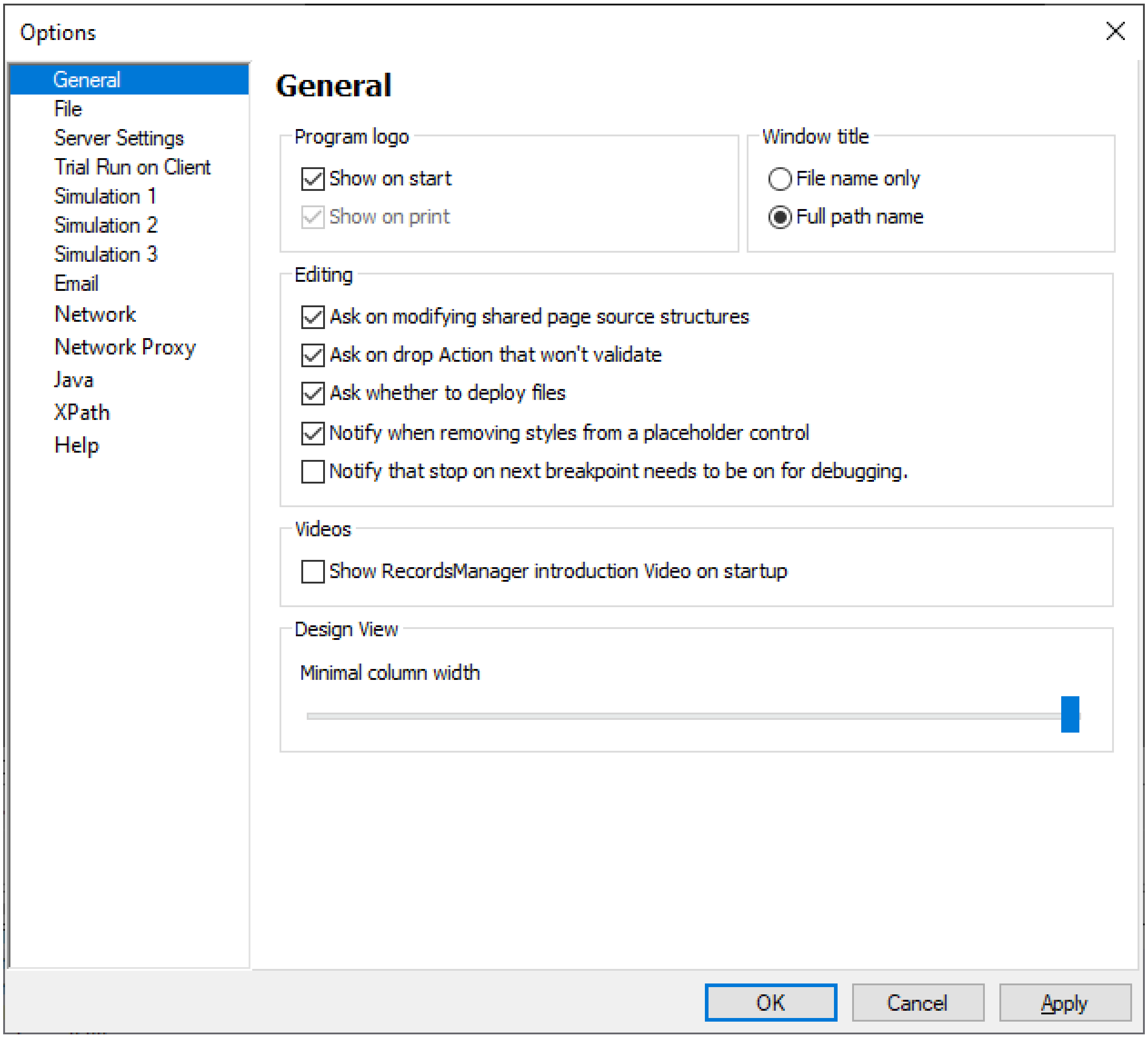
- *Show ScreenTips on toolbar:* Displays a popup when the mouse pointer is placed over an icon in any toolbar. The popup contains a short description of the icon function, as well as the associated keyboard shortcut, if one has been assigned and if the *Show shortcut keys* option has been checked .
- *Show shortcut keys in Screen Tips:* Defines whether shortcut information will be shown in screen tips.
- *Large icons:* Toggles the size of toolbar icons between standard and large.

## 28.10.5 Restore Toolbars and Windows

Shuts down MobileTogether Designer and restarts it with all toolbars and windows reset to the original state in which they were at installation.

## 28.10.6 Options

The **Options** command displays the Options dialog (*screenshot below*). The settings available in the various tabs are described in the sub-sections of this section.



Options

General

Program logo

☑ Show on start

☑ Show on print

Window title

○ File name only

● Full path name

Editing

☑ Ask on modifying shared page source structures

☑ Ask on drop Action that won't validate

☑ Ask whether to deploy files

☑ Notify when removing styles from a placeholder control

□ Notify that stop on next breakpoint needs to be on for debugging.

Videos

□ Show RecordsManager introduction Video on startup

Design View

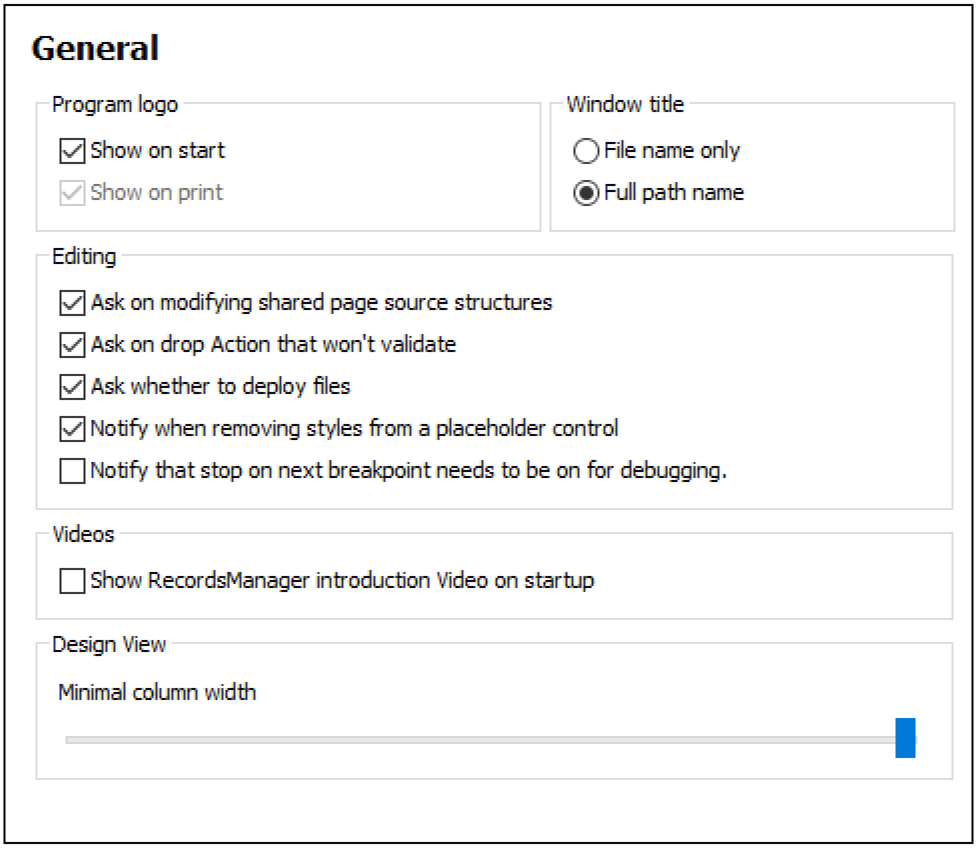
Minimal column width

The settings are organized into the following tabs:

- General 1665
- File 1666
- Server Settings 1666
- Trial Run on Client 1667
- Simulation 1 1668
- Simulation 2 1669
- Simulation 3 1673
- Email 1674
- Network 1675
- Network Proxy 1676
- Java 1678
- XPath 1678
- Help 1679

### 28.10.6.1 General

In the General tab (*screenshot below*) you can define the settings shown in the screenshot.



General

Program logo

☑ Show on start

☑ Show on print

Window title

○ File name only

● Full path name

Editing

☑ Ask on modifying shared page source structures

☑ Ask on drop Action that won't validate

☑ Ask whether to deploy files

☑ Notify when removing styles from a placeholder control

□ Notify that stop on next breakpoint needs to be on for debugging.

Videos

□ Show RecordsManager introduction Video on startup

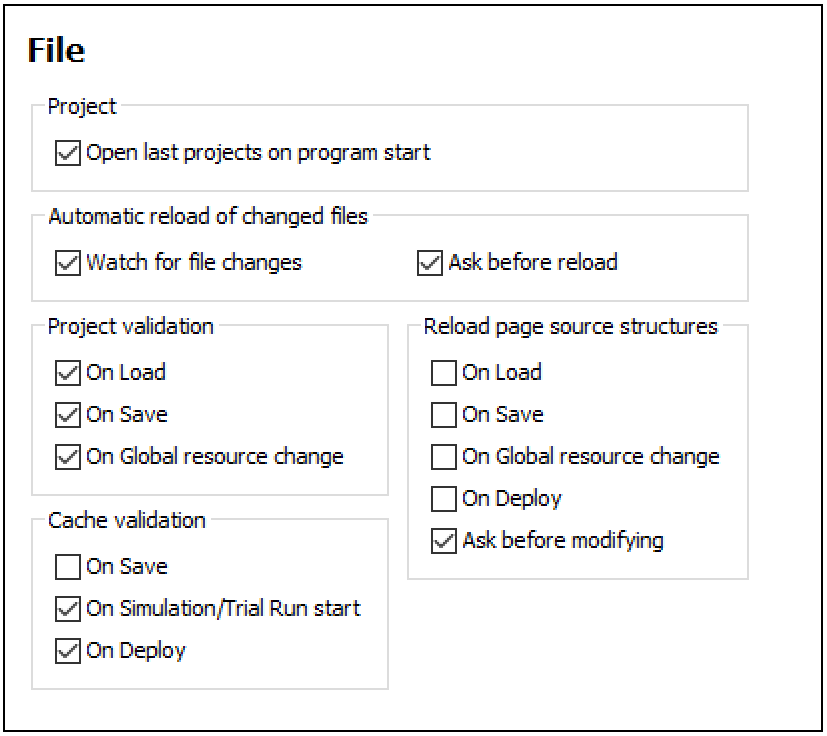
Design View

Minimal column width

- *Program logo:* Can be shown at program start and in print output.
- *Window title:* The application window can display either the file name only, or the full file path and file name.
- *Editing:* In situations where designer input is required, you are prompted about whether to go ahead with the action or not. For example, when a shared page resource is modified, you are asked whether the modifications should be available on all pages that share the resource, or whether the modifications should apply to the current page only. You can also switch on/off notifications related to placeholder controls 568 and the debugger 1388.
- *Videos:* An option to show a message about MobileTogether Designer demo videos when MobileTogether Designer is started with no design open. (To start MobileTogether Designer with no design open, close all designs and then close MobileTogether Designer.) The message contains a link to the demo video page on the Altova website. The videos on this page provide a quick introduction to the features of MobileTogether Designer.
- *Design View: Minimal column width:* Sets the minimal width of table columns in Design View. The slider provides options on a scale from `0` to `7`. You can drag the slider, or use the **Move Left** and **Move Right** keys. The selected value does not affect the width of table columns on client devices.

## 28.10.6.2 File

In the File tab (*screenshot below*) you can define the settings shown in the screenshot.



File

Project
☑ Open last projects on program start

Automatic reload of changed files
☑ Watch for file changes ☑ Ask before reload

Project validation
☑ On Load
☑ On Save
☑ On Global resource change

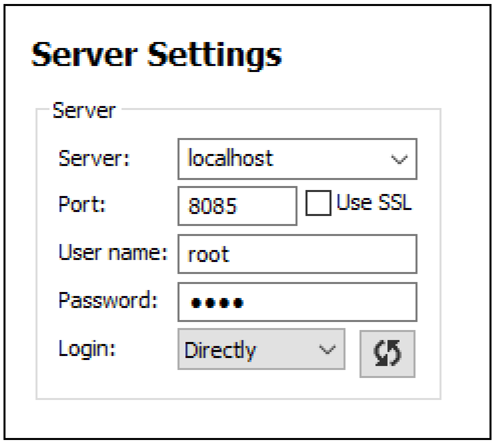
Reload page source structures
☐ On Load
☐ On Save
☐ On Global resource change
☐ On Deploy
☑ Ask before modifying

Cache validation
☐ On Save
☑ On Simulation/Trial Run start
☑ On Deploy

- *Project:* If selected, then on program start, all the projects are opened that were open when the program was last closed.
- *Reload of changed files:* Options to watch for changes made by another user and to ask whether to reload or not. If a file is reloaded, your own changes from the time of the last save will be lost.
- *Project validation:* When to carry out project validation. Select the options you want.
- *Reload page source structures:* When to reload page source structures. Select the options you want. The *Ask before modifying* option determines whether the user should be asked before modifying page source structures that are shared with one or more other pages. For example, if this option is selected and a shared page source structure is modified, then the user is asked to select from the following options: (i) whether the shared structure should be modified in all its occurrences (that is on all pages where it occurs), (ii) whether a copy of the data structure should be made with a different name for this page; this data structure can be modified subsequently without affecting the data structures on the other pages, (iii) whether to cancel the modification. To reload all page sources immediately, use the menu command **Project | Reload Page Source Structures** 1589 .
- *Cache validation:* Specifies when the cache is validated. Select the options you want.

## 28.10.6.3 Server Settings

In the Server Settings tab (*screenshot below*) you can define the connection and authentication settings of the MobileTogether Server to which you wish to connect MobileTogether Designer. These settings will be used when solutions are deployed to the server 1574 and when the server is used for workflow simulation 1619 . The user must have the corresponding MobileTogether Server rights: *Save workflow from designer* and *Run server simulation*. The access rights of users of MobileTogether Server are defined in the Web UI of MobileTogether Server. See the MobileTogether Server user manual for information about how to do this.



**Server Settings**

Server

Server: localhost

Port: 8085 □ Use SSL

User name: root

Password: ••••

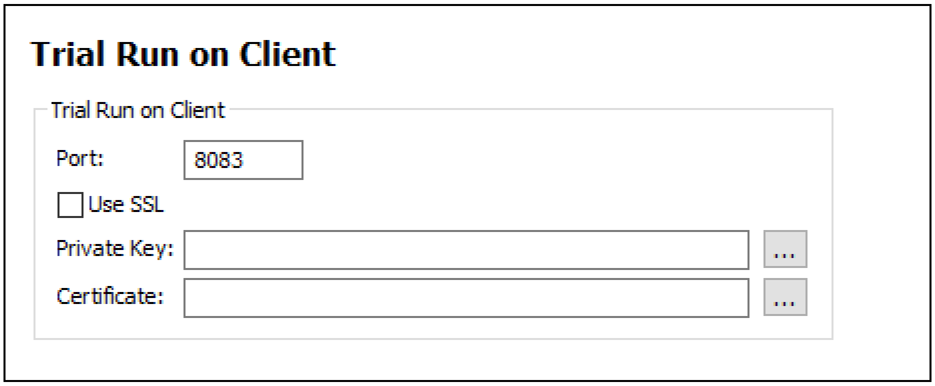
Login: Directly

If the Active Directory login feature of MobileTogether Server has been enabled for you as a domain user, then the login information you enter for connecting MobileTogether Designer to MobileTogether Server can be your domain authentication info. For example, if your Windows user name and password on your office network domain has been enabled for use as MobileTogether Server authentication, then you can enter your domain-specific user name and password.

To select whether user credentials directly specified in MobileTogether Server or domain-specific user credentials are to be used, select the appropriate option from the *Login* combo box (*see screenshot above*). The button next to the combo box is for updating the connection with MobileTogether Server.

## 28.10.6.4 Trial Run on Client

In the Trial Run on Client tab (*screenshot below*) you can define the local port via which MobileTogether Designer connects with the client. For the Trial Run on Client feature, MobileTogether Designer itself acts as the MobileTogether Server and serves the design and related data files directly to the client.



**Trial Run on Client**

Trial Run on Client

Port: 8083

□ Use SSL

Private Key:

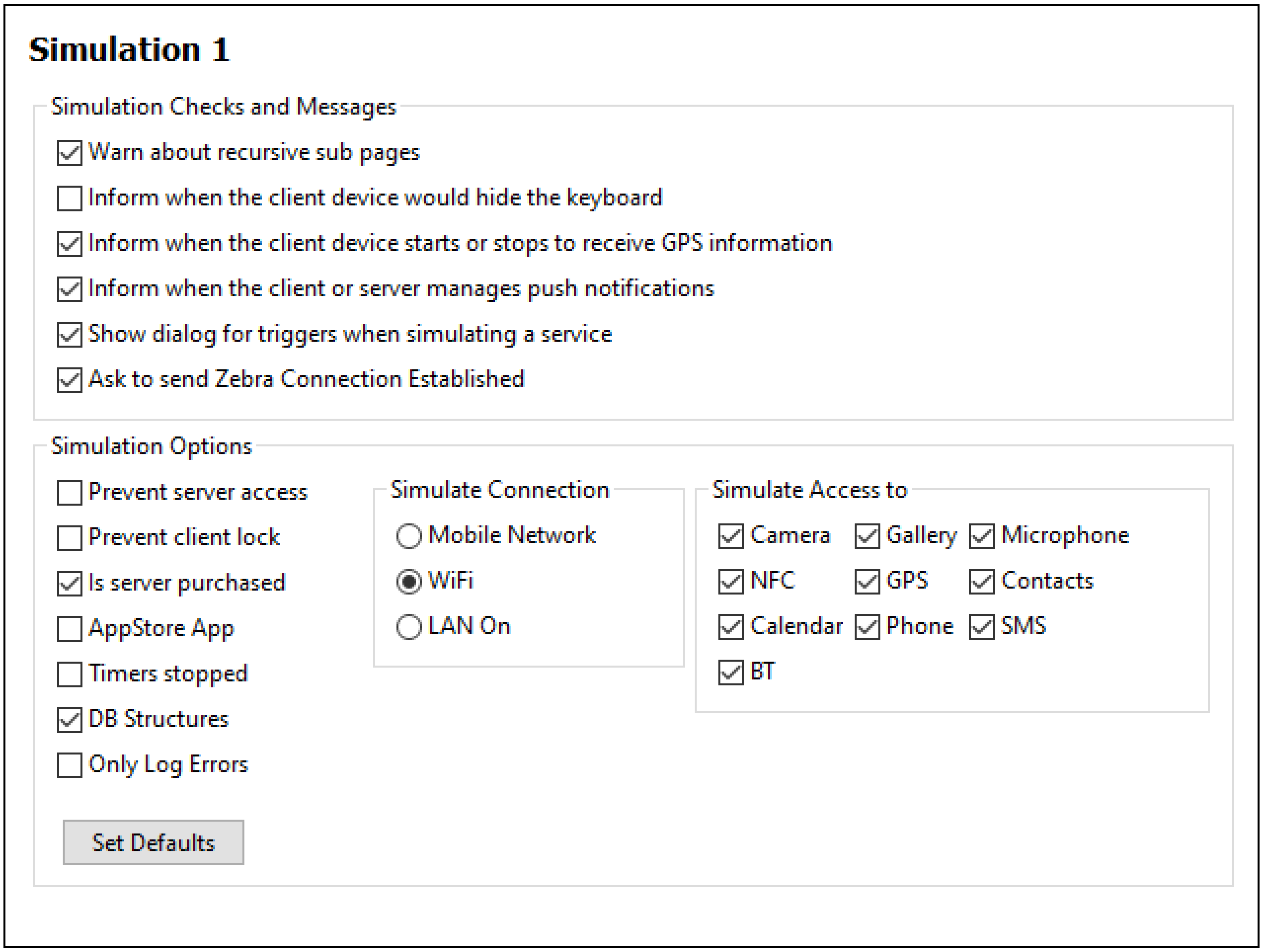
Certificate:

- *SSL:* Whether SSL is used.
- *Private key, certificate:* Browse for the SSL private key and certificate (if SSL is used).

### 28.10.6.5 Simulation 1

In the Simulation 1 tab (*screenshot below*), you can specify default aspects of simulations: simulation checks and messages, the simulation language, whether comment and processing instructions should be shown in the simulator's XML data tree, etc. Additional simulation options are available in the next tab, Simulation 2.

**Note:** You can change simulation options before running a simulation by clicking the **Simulator Options** button in the Simulator's toolbar[1356], and then changing settings as required. If you change a setting in the Simulator, that change is transmitted back to this Options pane and becomes the new default setting.



**Simulation 1**

Simulation Checks and Messages

☑ Warn about recursive sub pages
□ Inform when the client device would hide the keyboard
☑ Inform when the client device starts or stops to receive GPS information
☑ Inform when the client or server manages push notifications
☑ Show dialog for triggers when simulating a service
☑ Ask to send Zebra Connection Established

Simulation Options

□ Prevent server access
□ Prevent client lock
☑ Is server purchased
□ AppStore App
□ Timers stopped
☑ DB Structures
□ Only Log Errors

Simulate Connection

○ Mobile Network
● WiFi
○ LAN On

Simulate Access to

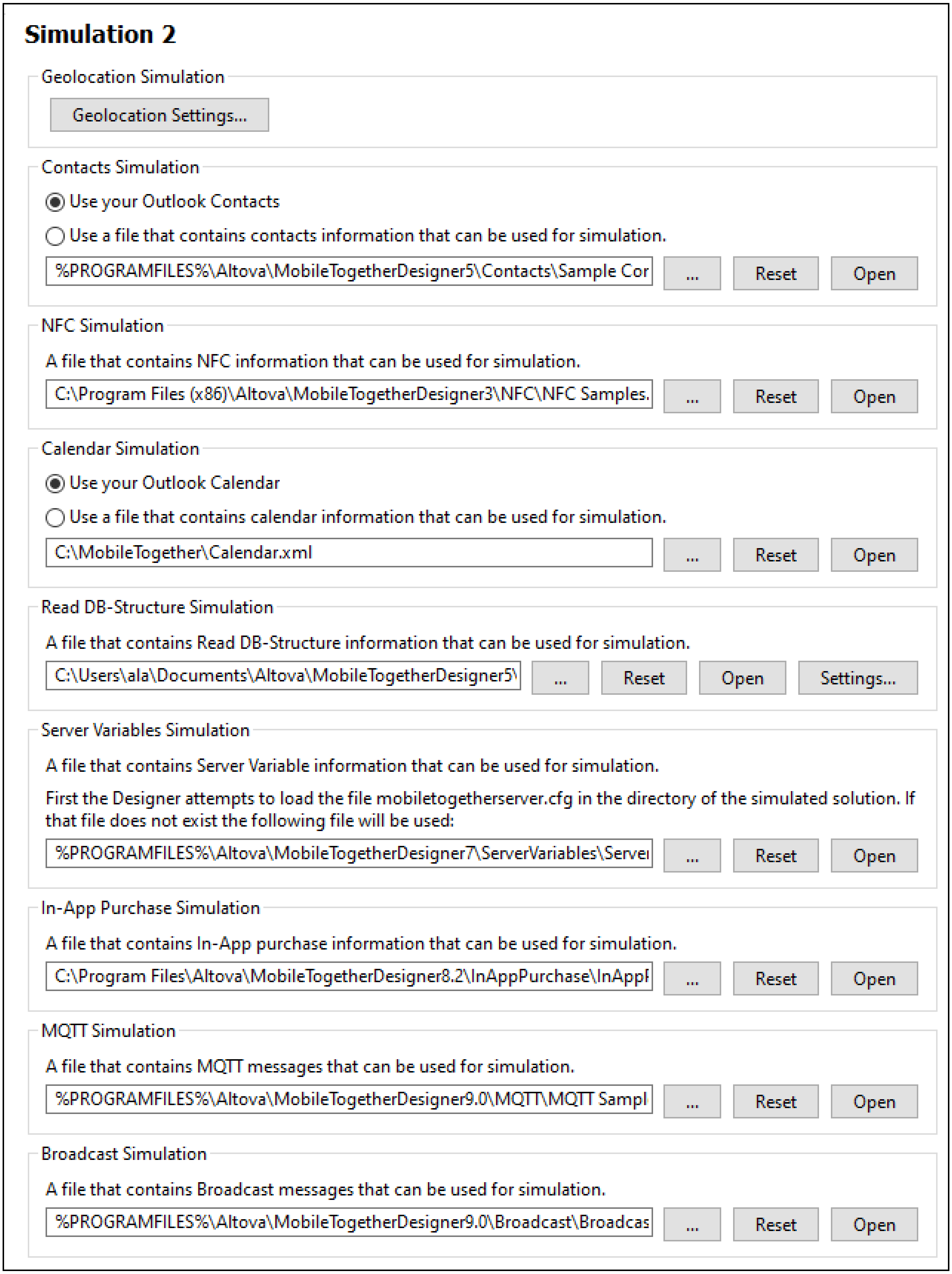
☑ Camera ☑ Gallery ☑ Microphone
☑ NFC ☑ GPS ☑ Contacts
☑ Calendar ☑ Phone ☑ SMS
☑ BT

- *Simulation checks and messages:* Specifies whether warnings should be issued about (i) recursive sub pages, (ii) the keyboard being hidden on the client, (iii) when the client starts or stops receiving GPS information, (iv) PN management, (v) whether a dialog to set service triggers[1383] should be displayed, (vi) whether to be prompted about sending a Zebra Connection Established message.
- *Options:* The settings you make here determine the default settings of the simulator[1356]. For example, you can set whether the simulator has access to the server or not, and whether a client lock[919] should be prevented. You can also specify whether a WiFi, LAN, or mobile network connection is simulated when the simulator starts. By default, the *Simulate WiFi* option is on. (In the Simulator[1356], you can

change the options at any time[1356].) The **Set Defaults** button resets the options to their original default settings. You can also simulate that certain client-side apps (like a camera or calendar) or functionality (such as Bluetooth (BT)) can be accessed. After the simulator is started, you can change these settings in the simulator's toolbar or its **Simulation** menu. To set default data sources for some of these simulated client-side apps, go to the Simulation 2[1669] tab in this dialog.

## 28.10.6.6 Simulation 2

In the Simulation 2 tab (*screenshot below*), you can specify simulation settings additional to those available in the Simulation 1 tab (*see previous point*). These settings include the definition of data sources for the simulation of client-side apps.



Simulation 2

Geolocation Simulation

Contacts Simulation

● Use your Outlook Contacts

○ Use a file that contains contacts information that can be used for simulation.

%PROGRAMFILES%\Altova\MobileTogetherDesigner5\Contacts\Sample Cor

NFC Simulation

A file that contains NFC information that can be used for simulation.

C:\Program Files (x86)\Altova\MobileTogetherDesigner3\NFC\NFC Samples.

Calendar Simulation

● Use your Outlook Calendar

○ Use a file that contains calendar information that can be used for simulation.

C:\MobileTogether\Calendar.xml

Read DB-Structure Simulation

A file that contains Read DB-Structure information that can be used for simulation.

C:\Users\ala\Documents\Altova\MobileTogetherDesigner5\

Server Variables Simulation

A file that contains Server Variable information that can be used for simulation.

First the Designer attempts to load the file mobiletogetherserver.cfg in the directory of the simulated solution. If that file does not exist the following file will be used:

%PROGRAMFILES%\Altova\MobileTogetherDesigner7\ServerVariables\Server

In-App Purchase Simulation

A file that contains In-App purchase information that can be used for simulation.

C:\Program Files\Altova\MobileTogetherDesigner8.2\InAppPurchase\InAppP

MQTT Simulation

A file that contains MQTT messages that can be used for simulation.

%PROGRAMFILES%\Altova\MobileTogetherDesigner9.0\MQTT\MQTT Sampl

Broadcast Simulation

A file that contains Broadcast messages that can be used for simulation.

%PROGRAMFILES%\Altova\MobileTogetherDesigner9.0\Broadcast\Broadcas

*Geolocation Simulation*
Enables default geolocation settings to be defined. See the section Geolocation Settings 1372 for details.

*Contacts Simulation*
You can specify whether to use your Microsoft Outlook* contacts to simulate the device's address book. If Outlook is not available, then a file can be used 1381.

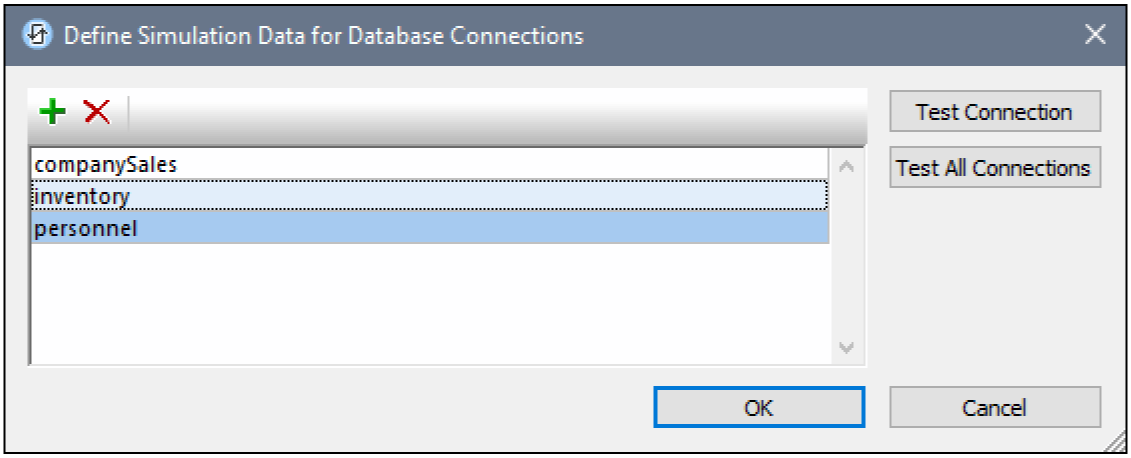
*NFC Simulation*
This setting specifies the file to use to simulate NFC tag discovery 1377.

*Calendar Simulation*
You can specify whether to use your Microsoft Outlook* calendar to simulate the device's calendar app. If Outlook is not available, then an XML file can be used to provide calendar simulation data 1382.

*Read DB-Structure Simulation*
Specifies an XML file that contains a connection to a DB that is to be read for simulations of the DB Read Structure 867 action. A dummy XML file, named **`Sample DB Read Structure.xml`**, is installed with MobileTogether Designer (version 5.0 and later) and is located in your Windows *"My Documents"* folder *(click link below to see folder location)*. Note that this XML file contains no DB connection data when it is installed. But you can quickly generate connection information for multiple databases and save this information in the XML file as described below. The DB connection that is used during a simulation will be the one that has the same name as that of the connection in the DB Read Structure 867 action. You can save the XML file to another location, or use some other XML file as the DB connection data file.

To generate one or more connections and save these to the currently selected XML file, do the following:

1. Click **Settings**.
2. In the dialog that appears (*see screenshot below*), click the **Add DB** button in the toolbar at top left.



3. In the DB Connection Wizard 961 that now appears, add a new DB connection by following the wizard's steps 961.
4. After the DB connection has been created, it will appear in the list of connections (*see screenshot above*).
5. Click **OK** to save *all* the connections to the XML file. If you wish to remove a connection before saving all connections, select that connection in the dialog and click the **Delete** button in the toolbar at top left (*see screenshot above*).

*Server Variables Simulation*
You can specify a server configuration file (`.cfg` file) that contains the server variables you want to simulate. Server variables would provide server-side information that can be accessed in the design via the **`mt-server-variables`** 1262 XPath extension function. In a server configuration file, the server variables are stored in the `[ServerVariables]` section of the file, as shown in the snippet below:

```
[ServerVariables]
Environment=Admin
Manual=AdminDocs
StartPage=Admin
```

For more information about the MobileTogether Server configuration file, see the MobileTogether Server user manual.

*In-App Purchase Simulation*
You can specify a file that provides data to simulate the product and purchase information that is available at app stores. See the topic about simulating in-app purchases in MobileTogether Designer 1516.

*MQTT Simulation*
You can specify the path to a file that contains data for simulating MQTT messages received. See the topic MQTT 1135 for details.

*Broadcast Simulation*
You can specify the path to a file that contains Broadcast messages for simulating Broadcast messages received. See the topic Broadcasts 1142 for details.

* *Microsoft Outlook is part of the Microsoft Office suite. Among other functionality, it provides a contacts manager and a calendar.*

## File paths in Windows

File paths given in this documentation will not be the same for all operating systems. You should note the following correspondences:

- (My) Documents folder: Located by default at the following locations. Example files are located in a sub-folder of this folder.

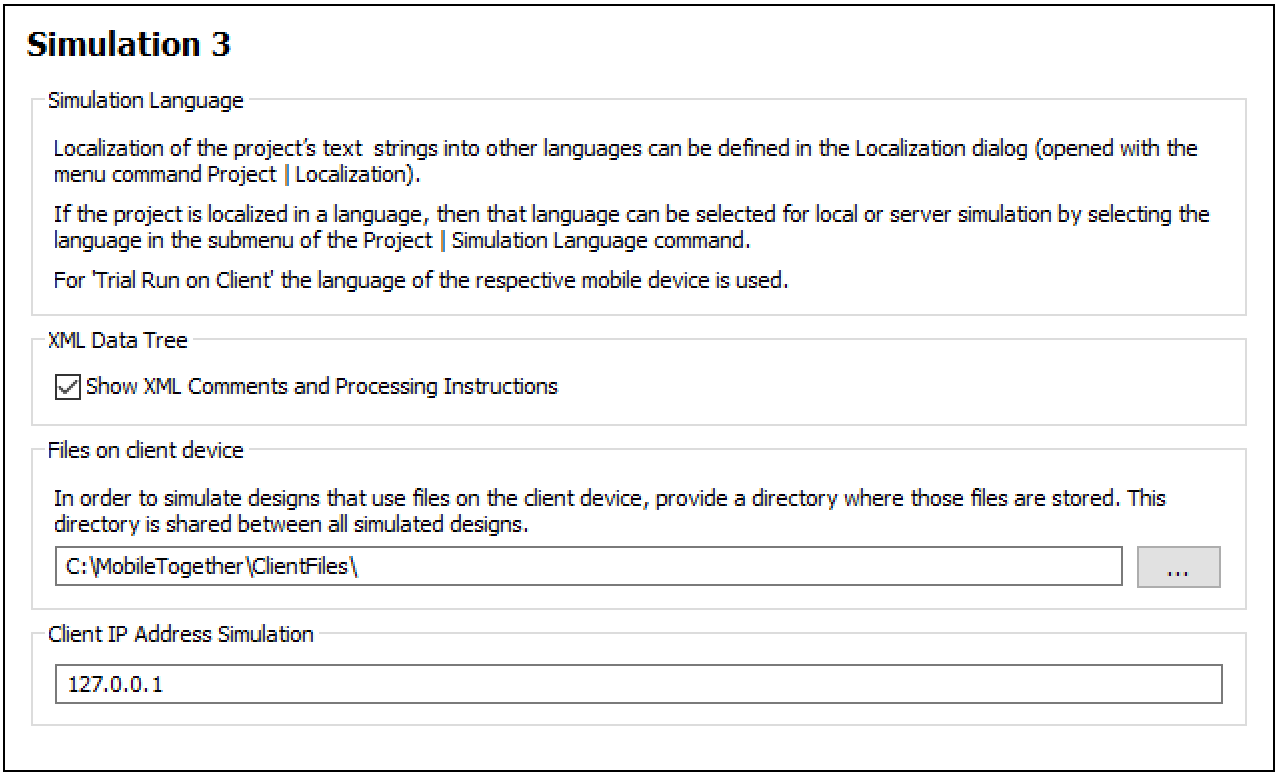
| | |
|---|---|
| Windows 7/8/10/11 | `C:\Users\<username>\Documents` |

- *Application folder:* The Application folder is the folder where your Altova application is located. The path to the Application folder is, by default, the following.

| | |
|---|---|
| Windows 7/8/10/11 | `C:\Program Files\Altova\` |
| 32-bit version on 64-bit OS | `C:\Program Files (x86)\Altova\` |

## 28.10.6.7 Simulation 3

In the Simulation 2 tab (*screenshot below*), you can specify simulation settings additional to those available in the Simulation 1 tab (*see previous point*). These settings include the definition of data sources for the simulation of client-side apps.



**Simulation 3**

Simulation Language

Localization of the project's text strings into other languages can be defined in the Localization dialog (opened with the menu command Project | Localization).

If the project is localized in a language, then that language can be selected for local or server simulation by selecting the language in the submenu of the Project | Simulation Language command.

For 'Trial Run on Client' the language of the respective mobile device is used.

XML Data Tree

☑ Show XML Comments and Processing Instructions

Files on client device

In order to simulate designs that use files on the client device, provide a directory where those files are stored. This directory is shared between all simulated designs.

C:\MobileTogether\ClientFiles\

Client IP Address Simulation

127.0.0.1

*Simulation Language:*
A hint about localization options. A project can be localized—that is, the text strings in the project can be translated—in the Localization dialog (**Project | Localization** 1600). If a project has been localized into some language, then that simulation language can be chosen as the simulation language in the submenu of the **Project | Simulation Language** 1606 command.

*XML Data Tree*
Specifies whether comments and processing instructions should also be displayed in the simulator's XML data tree.

*Files on client device:*
If the design references files on the client device, then these files will not be accessible during simulations. During simulations, the folder specified in this option will be looked up for client-side files. If client-side files are saved here with the same name as that with which they are referenced in the design, then they will be correctly accessed during simulations.

*Client IP Address Simulation*
A random text value can be used to simulate the IP address obtained by the **`mt-client-ip-address`** 1262 function.

## File paths in Windows

File paths given in this documentation will not be the same for all operating systems. You should note the following correspondences:

- (My) Documents folder: Located by default at the following locations. Example files are located in a sub-folder of this folder.

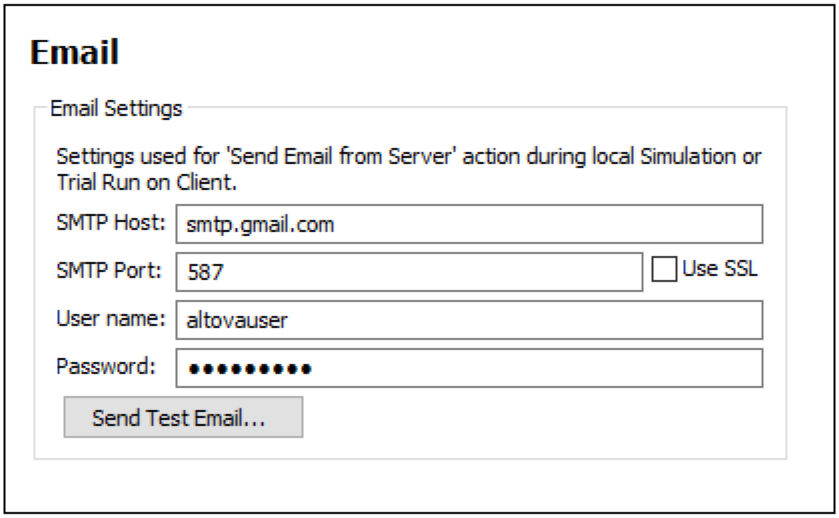
| | |
|---|---|
| Windows 7/8/10/11 | `C:\Users\<username>\Documents` |

- *Application folder:* The Application folder is the folder where your Altova application is located. The path to the Application folder is, by default, the following.

| | |
|---|---|
| Windows 7/8/10/11 | `C:\Program Files\Altova\` |
| 32-bit version on 64-bit OS | `C:\Program Files (x86)\Altova\` |

## 28.10.6.8 Email

The settings in the Email tab are used during local simulations 1355 for access to the SMTP server of an email service provider (usually your ISP). They are used by the Send Email (from Server) 693 action, which enables emails to be sent by the end user via the server. In a live, real-time scenario, the settings to access the SMTP server are configured in MobileTogether Server. During local simulations, however, the SMTP server information is not available (because MobileTogether Server is not accessed during local simulations). SMTP server settings for local simulations are therefore entered in this tab (*screenshot below*).



Email

Email Settings

Settings used for 'Send Email from Server' action during local Simulation or Trial Run on Client.

SMTP Host: smtp.gmail.com

SMTP Port: 587 □ Use SSL

User name: altovauser

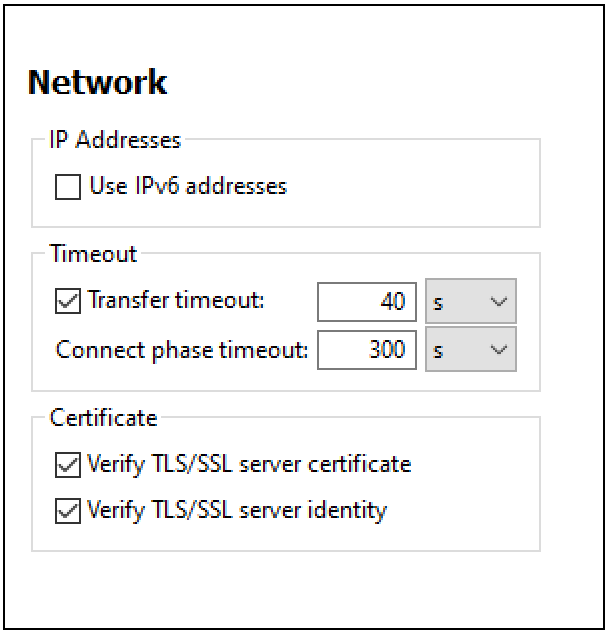
Password: ●●●●●●●●●●

- *SMTP Host and SMTP Port:* These are the SMTP host name and SMTP port of your ISP's SMTP server. These details are provided to you by your ISP.
- *User Name and Password:* The user name and password of an email account that is registered with the email service provider.

After entering the details, click **OK**. You can send a test email to check whether the settings work correctly.

## 28.10.6.9 Network

The **Network** section (*screenshot below*) enables you to configure important network settings.



Network

IP Addresses

☐ Use IPv6 addresses

Timeout

☑ Transfer timeout: 40 s

Connect phase timeout: 300 s

Certificate

☑ Verify TLS/SSL server certificate

☑ Verify TLS/SSL server identity

*IP addresses*
When host names resolve to more than one address in mixed IPv4/IPv6 networks, selecting this option causes the IPv6 addresses to be used. If the option is not selected in such environments and IPv4 addresses are available, then IPv4 addresses are used.

*Timeout*

- *Transfer timeout:* If this limit is reached for the transfer of any two consecutive data packages of a transfer (sent or received), then the entire transfer is aborted. Values can be specified in seconds [s] or milliseconds [ms], with the default being 40 seconds. If the option is not selected, then there is no time limit for aborting a transfer.
- *Connection phase timeout:* This is the time limit within which the connection has to be established, including the time taken for security handshakes. Values can be specified in seconds [s] or milliseconds [ms], with the default being 300 seconds. This timeout cannot be disabled.

*Certificate*

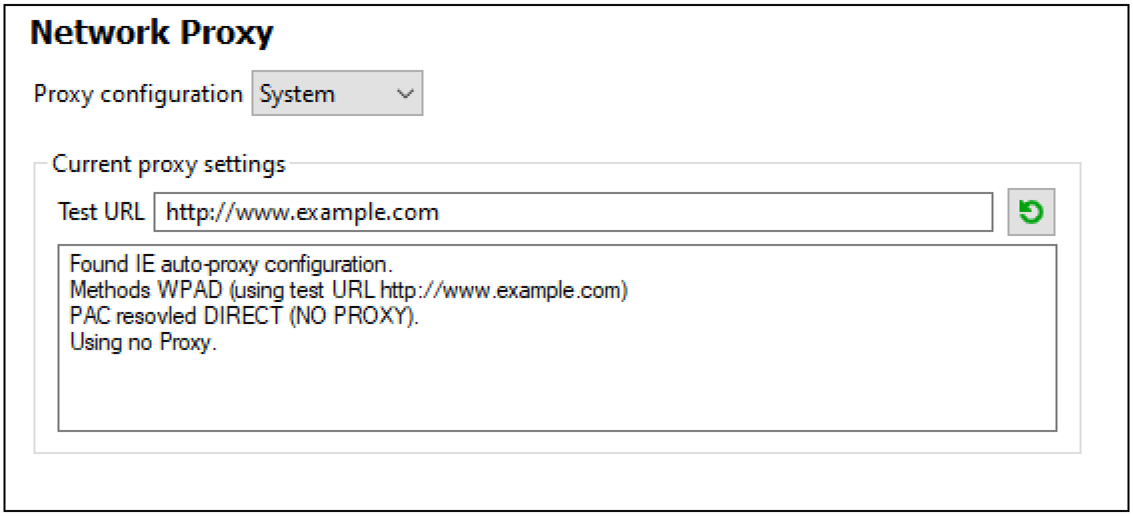
- *Verify TLS/SSL server certificate:* If selected, then the authenticity of the server's certificate is checked by verifying the chain of digital signatures until a trusted root certificate is reached. This option is enabled by default. If this option is not selected, then the communication is insecure, and attacks (for example, a man-in-the-middle attack) would not be detected. Note that this option does not verify that the certificate is actually for the server that is communicated with. To enable full security, both the certificate and the identity must be checked (*see next option*).
- *Verify TLS/SSL server identity:* If selected, then the server's certificate is verified to belong to the server we intend to communicate with. This is done by checking that the server name in the URL is the same

as the name in the certificate. This option is enabled by default. If this option is not selected, then the server's identify is not checked. Note that this option does not enable verification of the server's certificate. To enable full security, both the certificate as well as the identity must be checked (*see previous option*).

## 28.10.6.10 Network Proxy

The *Network Proxy* section enables you to configure custom proxy settings. These settings affect how the application connects to the Internet (for XML validation purposes, for example). By default, the application uses the system's proxy settings, so you should not need to change the proxy settings in most cases. If necessary, however, you can set an alternative network proxy by selecting, in the *Proxy Configuration* combo box, either *Automatic* or *Manual* to configure the settings accordingly.

**Note:** The network proxy settings are shared among all Altova MissionKit applications. So, if you change the settings in one application, all MissionKit applications will be affected.



Proxy configuration: System

Current proxy settings

Test URL: http://www.example.com

```
Found IE auto-proxy configuration.
Methods WPAD (using test URL http://www.example.com)
PAC resovled DIRECT (NO PROXY).
Using no Proxy.
```

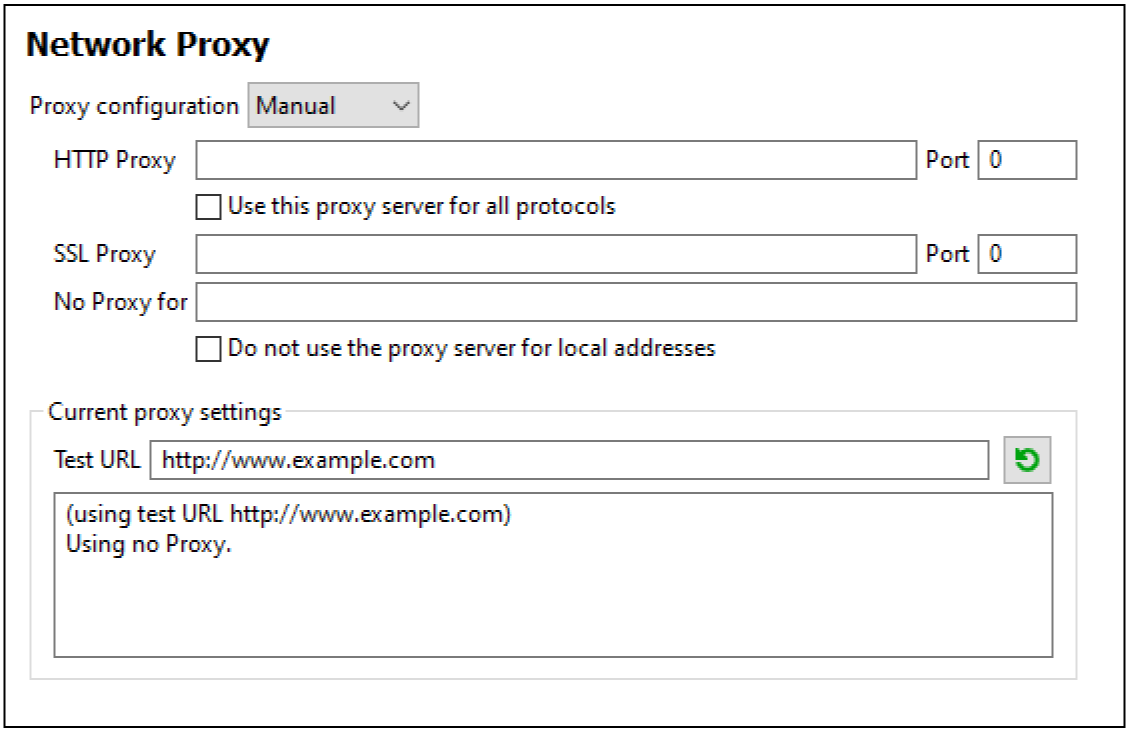
*Use system proxy settings*
Uses the Internet Explorer (IE) settings configurable via the system proxy settings. It also queries the settings configured with **`netsh.exe winhttp`**.

*Automatic proxy configuration*
The following options are provided:

- *Auto-detect settings:* Looks up a WPAD script (**`http://wpad.LOCALDOMAIN/wpad.dat`**) via DHCP or DNS, and uses this script for proxy setup.
- *Script URL:* Specify an HTTP URL to a proxy-auto-configuration (**`.pac`**) script that is to be used for proxy setup.
- *Reload:* Resets and reloads the current auto-proxy-configuration. This action requires Windows 8 or newer, and may need up to 30s to take effect.

*Manual proxy configuration*
Manually specify the fully qualified host name and port for the proxies of the respective protocols. A supported scheme may be included in the host name (for example: **`http://hostname`**). It is not required that the scheme

is the same as the respective protocol if the proxy supports the scheme.



Network Proxy

Proxy configuration: Manual

HTTP Proxy: Port: 0

□ Use this proxy server for all protocols

SSL Proxy: Port: 0

No Proxy for:

□ Do not use the proxy server for local addresses

Current proxy settings

Test URL: http://www.example.com

```
(using test URL http://www.example.com)
Using no Proxy.
```

The following options are provided:

- *HTTP Proxy:* Uses the specified host name and port for the HTTP protocol. If *Use this proxy server for all protocols* is selected, then the specified HTTP proxy is used for all protocols.
- *SSL Proxy:* Uses the specified host name and port for the SSL protocol.
- *No Proxy for:* A semi-colon (`;`) separated list of fully qualified host names, domain names, or IP addresses for hosts that should be used without a proxy. IP addresses may not be truncated and IPv6 addresses have to be enclosed by square brackets (for example: `[2606:2800:220:1:248:1893:25c8:1946]`). Domain names must start with a leading dot (for example: `.example.com`).
- *Do not use the proxy server for local addresses:* If checked, adds **`<local>`** to the *No Proxy for* list. If this option is selected, then the following will not use the proxy: (i) **`127.0.0.1`**, (ii) **`[::1]`**, (iii) all host names not containing a dot character (`.`).

*Current proxy settings*

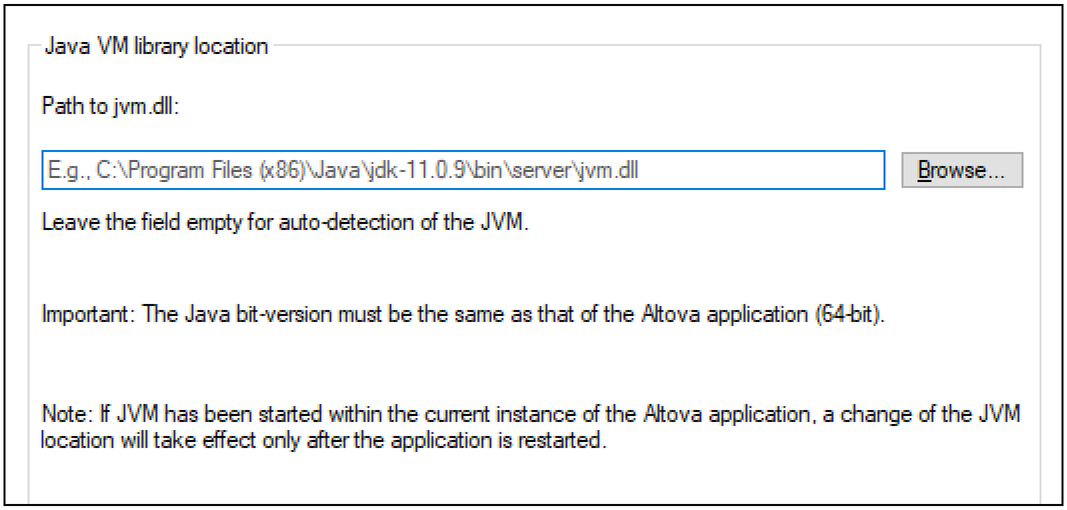
Provides a verbose log of the proxy detection. It can be refreshed with the **Refresh** button to the right of the *Test URL* field (for example, when changing the test URL, or when the proxy settings have been changed).

- *Test URL:* A test URL can be used to see which proxy is used for that specific URL. No I/O is done with this URL. This field must not be empty if proxy-auto-configuration is used (either through *Use system proxy settings* or *Authomatic proxy configuration*).

## 28.10.6.11 Java

In the *Java* section (*see screenshot below*), you can optionally enter the path to a Java VM (Virtual Machine) on your file system. Note that adding a custom Java VM path is not always necessary. By default, MobileTogether Designer attempts to detect the Java VM path automatically by reading (in this order) the Windows registry and the JAVA_HOME environment variable. The custom path added in this dialog box will take priority over any other Java VM path detected automatically.

You may need to add a custom Java VM path, for example, if you are using a Java virtual machine which does not have an installer and does not create registry entries (e.g., Oracle's OpenJDK). You might also want to set this path if you need to override, for whatever reason, any Java VM path detected automatically by MobileTogether Designer.


Java VM library location

Path to jvm.dll:

E.g., C:\Program Files (x86)\Java\jdk-11.0.9\bin\server\jvm.dll

Leave the field empty for auto-detection of the JVM.

Important: The Java bit-version must be the same as that of the Altova application (64-bit).

Note: If JVM has been started within the current instance of the Altova application, a change of the JVM location will take effect only after the application is restarted.

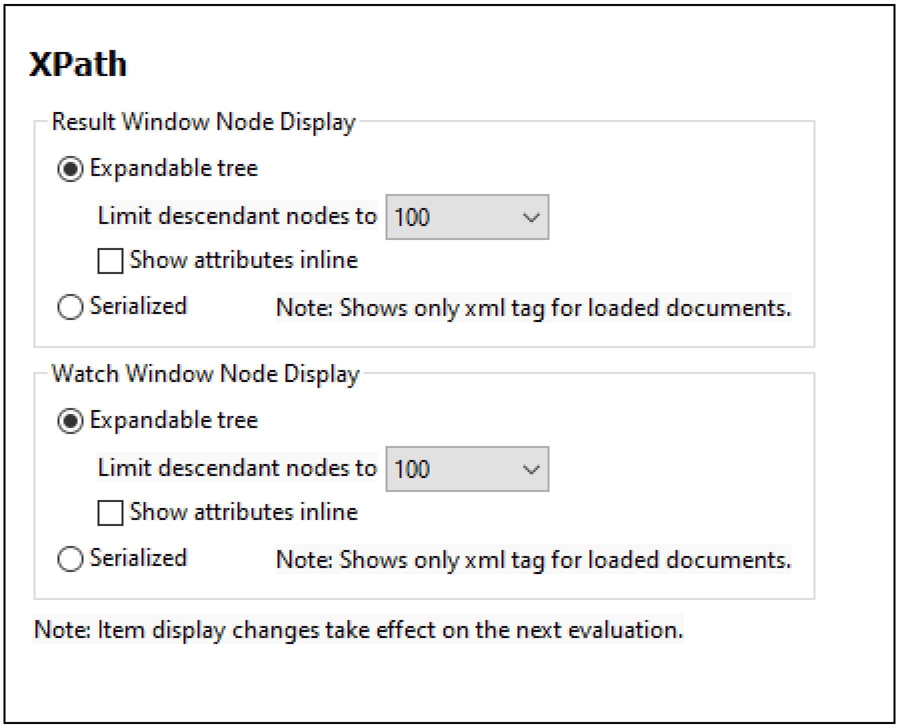
Note the following:

- The Java VM path is shared between Altova desktop (not server) applications. Consequently, if you change it in one application, it will automatically apply to all other Altova applications.
- The path must point to the `jvm.dll` file from the `\bin\server` or `\bin\client` directory, relative to the directory where the JDK was installed.
- The MobileTogether Designer platform (32-bit, 64-bit) must be the same as that of the JDK.
- After changing the Java VM path, you may need to restart MobileTogether Designer for the new settings to take effect.

Changing the Java VM path affects database connectivity via JDBC.

## 28.10.6.12 XPath

The XPath options (*screenshot below*) enable you to set the display options of results in XPath Debugger[1252], separately for the main results and for the Watch window.



**XPath**

Result Window Node Display

● Expandable tree

Limit descendant nodes to 100

□ Show attributes inline

○ Serialized Note: Shows only xml tag for loaded documents.

Watch Window Node Display

● Expandable tree

Limit descendant nodes to 100

□ Show attributes inline

○ Serialized Note: Shows only xml tag for loaded documents.

Note: Item display changes take effect on the next evaluation.

- *Expandable tree or serialized:* Select whether the results should be shown as trees with descendants, or as serialized XML nodes. Note that serialization works only for documents loaded for the evaluation.
- *Descendant nodes:* Sets the maximum of descendant nodes to show.
- *Show attributes inline:* Attributes are additionally shown on the same line as the element. The benefit of this is that you can see the attributes even when the element node is colapsed.

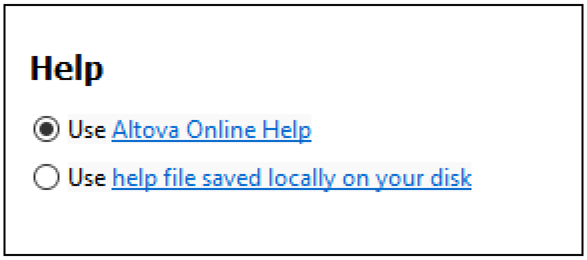
After entering the details, click **OK**. You can send a test email to check whether the settings work correctly.

## 28.10.6.13 Help

MobileTogether Designer provides Help (the user manual) in two formats:

- Online Help, in HTML format, which is available at the Altova website. In order to access the Online Help you will need Internet access.
- A Help file in PDF format, which is installed on your machine when you install MobileTogether Designer. It is named `MobileTogether Designer.pdf` and is located in the application folder (in the Program Files folder). If you do not have Internet access, you can always open this locally saved Help fie.

The Help option (*screenshot below*) enables you to select which of the two formats is opened when you click the **Help (F1)** command in the **Help** menu.



Help

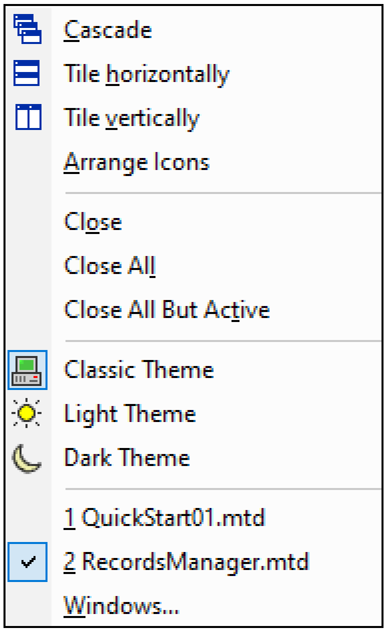
● Use Altova Online Help

○ Use help file saved locally on your disk

You can change this option at any time for the new selection to take effect. The links in this section (*see screenshot above*) open the respective Help format.

# 28.11 Window

The **Window** menu contains commands that let you organize individual application and document windows within the GUI. You can cascade or tile open document windows, and you can change the theme of the interface.



## Cascade, Tile Horizontally/Vertically

The **Cascade** command arranges document windows so that they are staggered in a sequence from back to forward.

The **Tile Horizontally** and **Tile Vertically** arranges the windows of open and non-minimized documents so that they are re-sized as tiles that are all visible within the application window.

## Close commands

The Close commands provide options to quickly close one of different sets of open files. If any document in the set you are closing has been modified, then you will be prompted about whether you want to save the modified file before closing it.

## Themes

MobileTogether Designer offers you a choice of the three themes listed below. When you select a theme, it is applied immediately.

- Classic (the default)
- Light
- Dark

Themes for simulations can be set in the Simulation window[1355].

## Currently open window list

This list shows all currently open windows, and lets you quickly switch between them. You can also use **CTRL+F6** keyboard shortcuts to cycle through the open windows.

# 28.12 Help

The **Help** menu contains commands required to get help or more information about MobileTogether Designer, as well as links to information and support pages on the Altova web server.

The **Help** menu contains the following commands:

- Help 1683
- Software Activation 1683
- Order Form 1683
- Registration 1683
- Check for Updates 1683
- Support Center 1685
- Show Video Demos 1685
- RecordsManager on the Internet 1684
- RecordsManager Videos 1684
- MobileTogether Designer on the Internet 1685
- About MobileTogether Designer 1685

## 28.12.1 Help

The **Help (F1)** command opens the application's Help documentation (its user manual). By default, the Online Help in HTML format at the Altova website will be opened.

If you do not have Internet access or do not want, for some other reason, to access the Online Help, you can use the locally stored version of the user manual. The local version is a PDF file named `MobileTogether Designer.pdf` that is stored in the application folder (in the Program Files folder).

If you want to change the default format to open (Online Help or local PDF), do this in the Help section of the Options dialog (menu command **Tools | Options**).

## 28.12.2 Activation, Order Form, Registration, Updates

⊟ Software Activation

*License your product*
After you download your Altova product software, you can license—or activate—it using either a free evaluation key or a purchased permanent license key.

- ***Free permanent license.*** When you first start the software after downloading and installing it, the **Software Activation** dialog will pop up. In it is a button to request a free permanent license. Click it to get your license. When you click this button, your machine-ID will be hashed and sent to Altova via HTTPS. The license information will be sent back to the machine via an HTTP response. If the license is created successfully, a dialog to this effect will appear in your Altova application. On clicking **OK** in this dialog, the software will be activated **on this particular machine**.

**Note:** For multi-user licenses, each user will be prompted to enter his or her own name.

*Your license email and the different ways to license (activate) your Altova product*
The license email that you receive from Altova will contain your license file as an attachment. The license file has a **`.altova_licenses`** file extension.

To activate your Altova product, you can do one of the following:

- Save the license file (**`.altova_licenses`**) to a suitable location, double-click the license file, enter any requested details in the dialog that appears, and finish by clicking **Apply Keys**.
- Save the license file (**`.altova_licenses`**) to a suitable location. In your Altova product, select the menu command **Help | Software Activation**, and then **Upload a New License**. Browse for or enter the path to the license file, and click **OK**.

You can access the **Software Activation** dialog at any time by clicking the **Help | Software Activation** command.

⊟ Order Form

The **Order Form** command takes you to the secure Altova Online Shop, where you can purchase license keys for Altova products. Altova MobileTogether Designer, however, is a free product and does not need to be ordered. You can request a free permanent license directly from within the program, via the Software Activation dialog *(see above)*.

⊟ Registration

Opens the Altova Product Registration page in a tab of your browser. Registering your Altova software will help ensure that you are always kept up to date with the latest product information.

⊟ Check for Updates

Checks with the Altova server whether a newer version than yours is currently available and displays a message accordingly.

## 28.12.3 RecordsManager Information

▼ RecordsManager on the Internet

⊟ *Description*

A link to the RecordsManager page at the Altova website. Click this link to go to learn more about RecordsManager.

▼ RecordsManager Videos

⊟ *Description*

A link to RecordsManager videos at the Altova website. These videos show you how to use the features of RecordsManager.

## 28.12.4 Other Commands

▼ Support Center

⊟ *Description*

A link to the MobileTogether Support Forum on the Internet. The Support Forums provides discussion forums, where problems are discussed, and announcements related to MobileTogether.

▼ Show Video Demos

⊟ *Description*

A link to the MobileTogether Designer video demo page on the Altova website. The videos on this page show you how to get started with using MobileTogether Designer.

▼ MobileTogether Designer on the Internet

⊟ *Description*

A link to the Altova website on the Internet. You can learn more about MobileTogether Designer and related technologies and products at the Altova website.

▼ About MobileTogether Designer

⊟ *Description*

Displays the splash window and version number of your product.

# 29 Frequently Asked Questions

*My project uses MySQL over ODBC. If I deploy my solution to MobileTogether Server and run it from there, I get the following error: "Database: [Microsoft][ODBC Driver Manager] Data source name not found and no default driver specified... Retrieval from database resulted in error." How do I resolve this?*

You have to consider that MobileTogether Server is installed as a Windows Service, and therefore by default doesn't run under your Windows account. Try these options to resolve your issue:

- Your Data Source Name (DSN) is probably defined as a User DSN. Use Windows Control Panel to move/recreate it as a System DSN. Then, if necessary, redo the connection in MobileTogether Designer, and redeploy.
- Assign the MobileTogether Server service to your Windows user account. After installation of MobileTogether Server, use Windows Services app to assign your user account.
- Use MobileTogether's Global Resources mechanism[1334] to use a different connection, or even a different database, for MobileTogether Server and MobileTogether Designer.

*I am using an ODBC connection with a Sybase DB (Sybase ASE ODBC Driver 4.10.00.00). It looks like the user name and password are not being stored to the registry or file DSN. Why?*

If you create a data source using the ODBC-Administrator, user name and password are never stored to the registry or the file DSN. In order to make the connection work for MobileTogether Server, which of course cannot show a popup to enter username and password, you have to add this information manually. This needs to be done for System DSNs, UserDSNs (as REG_SZ) and FileDSNs:

- `Name: UID     Value: <Your UserID>`
- `Name: PWD     Value: <Your Password>`

*If I use a web browser as a client, where is the persistent data stored?*

Data that is saved on a web client is saved in the local storage (aka web storage) of your browser. HTML 5.0 local storage is supported in the following browsers:

| | | | | | | |
|---|---|---|---|---|---|---|
| IE 8.0+ | Firefox 3.5+ | Safari 4.0+ | Chrome 4.0+ | Opera 10.5+ | iPhone 2.0+ | Android 2.0+ |

# 30 Appendices

Information included in this section:

- XSLT and XPath/XQuery Functions 1688
- License Information 1767

# 30.1 XSLT and XPath/XQuery Functions

This section lists Altova extension functions that can be used in XPath and/or XQuery expressions. Altova extension functions can be used with Altova's XSLT and XQuery engines, and provide functionality additional to that available in the function libraries defined in the W3C standards.

This section describes XPath/XQuery extension functions that have been created by Altova to provide additional operations. These extension functions 1689 can be computed by Altova's XSLT and XQuery engines according to the rules described in this section. For information about the regular XPath/XQuery functions, see Altova's XPath/XQuery Function Reference.

## General points

The following general points should be noted:

- Functions from the core function libraries defined in the W3C specifications can be called without a prefix. That's because the Altova XSLT and XQuery engines read non-prefixed functions as belonging to the namespace `http://www.w3.org/2005/xpath-functions`, which is the default functions namespace specified in the XPath/XQuery functions specifications. If this namespace is explicitly declared in an XSLT or XQuery document, the prefix used in the namespace declaration can also optionally be used on function names.
- In general, if a function expects a sequence of one item as an argument, and a sequence of more than one item is submitted, then an error is returned.
- All string comparisons are done using the Unicode codepoint collation.
- Results that are QNames are serialized in the form `[prefix:]localname`.

*Precision of xs:decimal*

The precision refers to the number of digits in the number, and a minimum of 18 digits is required by the specification. For division operations that produce a result of type `xs:decimal`, the precision is 19 digits after the decimal point with no rounding.

*Implicit timezone*

When two `date`, `time`, or `dateTime` values need to be compared, the timezones of the values being compared need to be known. When the timezone is not explicitly given in such a value, the implicit timezone is used. The implicit timezone is taken from the system clock, and its value can be checked with the `implicit-timezone()` function.

*Collations*

The default collation is the Unicode codepoint collation, which compares strings on the basis of their Unicode codepoint. The engine uses the Unicode Collation Algorithm. Other supported collations are the ICU collations listed below; to use one of these, supply its URI as given in the table below. Any string comparisons, including for the `max` and `min` functions, will be made according to the specified collation. If the collation option is not specified, the default Unicode-codepoint collation is used.

| Language | URIs |
|---|---|
| `da`: Danish | `da_DK` |

| | |
|---|---|
| de: German | de_AT, de_BE, de_CH, de_DE, de_LI, de_LU |
| en: English | en_AS, en_AU, en_BB, en_BE, en_BM, en_BW, en_BZ, en_CA, en_GB, en_GU, en_HK, en_IE, en_IN, en_JM, en_MH, en_MP, en_MT, en_MU, en_NA, en_NZ, en_PH, en_PK, en_SG, en_TT, en_UM, en_US, en_VI, en_ZA, en_ZW |
| es: Spanish | es_419, es_AR, es_BO, es_CL, es_CO, es_CR, es_DO, es_EC, es_ES, es_GQ, es_GT, es_HN, es_MX, es_NI, es_PA, es_PE, es_PR, es_PY, es_SV, es_US, es_UY, es_VE |
| fr: French | fr_BE, fr_BF, fr_BI, fr_BJ, fr_BL, fr_CA, fr_CD, fr_CF, fr_CG, fr_CH, fr_CI, fr_CM, fr_DJ, fr_FR, fr_GA, fr_GN, fr_GP, fr_GQ, fr_KM, fr_LU, fr_MC, fr_MF, fr_MG, fr_ML, fr_MQ, fr_NE, fr_RE, fr_RW, fr_SN, fr_TD, fr_TG |
| it: Italian | it_CH, it_IT |
| ja: Japanese | ja_JP |
| nb: Norwegian Bokmal | nb_NO |
| nl: Dutch | nl_AW, nl_BE, nl_NL |
| nn: Nynorsk | nn_NO |
| pt: Portuguese | pt_AO, pt_BR, pt_GW, pt_MZ, pt_PT, pt_ST |
| ru: Russian | ru_MD, ru_RU, ru_UA |
| sv: Swedish | sv_FI, sv_SE |

*Namespace axis*

The namespace axis is deprecated in XPath 2.0. Use of the namespace axis is, however, supported. To access namespace information with XPath 2.0 mechanisms, use the `in-scope-prefixes()`, `namespace-uri()` and `namespace-uri-for-prefix()` functions.

## 30.1.1 Altova Extension Functions

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | XP1 XP2 XP3.1 |
| *XSLT functions (used in XPath expressions in XSLT):* | XSLT1 XSLT2 XSLT3 |
| *XQuery functions (used in XQuery expressions in XQuery):* | XQ1 XQ3.1 |

## Usage of Altova extension functions

In order to use Altova extension functions, you must declare the Altova extension functions namespace (*first highlight in code listing below*) and then use the extension functions so that they are resolved as belonging to this namespace (*see second highlight*). The example below uses the Altova extension function named `age`.

```
<xsl:stylesheet version="2.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:fn="http://www.w3.org/2005/xpath-functions"
   xmlns:altova="http://www.altova.com/xslt-extensions">
   <xsl:output method="text" encoding="ISO-8859-1"/>
   <xsl:template match="Persons">
      <xsl:for-each select="Person">
         <xsl:value-of select="concat(Name, ': ')"/>
         <xsl:value-of select="altova:age(xs:date(BirthDate))"/>
         <xsl:value-of select="' years&#x0A;'"/>
      </xsl:for-each>
   </xsl:template>
</xsl:stylesheet>
```

## XPath/XQuery functions

XPath/XQuery functions can be used both in XPath expressions as well as in XQuery expressions:

- Date/Time 1690
- Geolocation 1707
- Image-related 1718
- Numeric 1727
- Schema 1730
- Sequence 1749
- String 1758

# 30.1.1.1 XPath/XQuery Functions: Date and Time

Altova's date/time extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data held as XML Schema's various date and time datatypes.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | XP1 XP2 XP3.1 |
| *XSLT functions (used in XPath expressions in XSLT):* | XSLT1 XSLT2 XSLT3 |
| *XQuery functions (used in XQuery expressions in XQuery):* | XQ1 XQ3.1 |

▼ Grouped by functionality

- Add a duration to xs:dateTime and return xs:dateTime 1692
- Add a duration to xs:date and return xs:date 1694
- Add a duration to xs:time and return xs:time 1695
- Format and retrieve durations 1694
- Remove timezone from functions that generate current date/time 1696
- Return days, hours, minutes, and seconds from durations 1697
- Return weekday as integer from date 1699
- Return week number as integer from date 1699
- Build date, time, or duration type from lexical components of each type 1701
- Construct date, dateTime, or time type from string input 1702
- Age-related functions 1704
- Epoch time (Unix time) functions 1705

▼ Listed alphabetically

`altova:add-days-to-date` 1694
`altova:add-days-to-dateTime` 1692
`altova:add-hours-to-dateTime` 1692
`altova:add-hours-to-time` 1695
`altova:add-minutes-to-dateTime` 1692
`altova:add-minutes-to-time` 1695
`altova:add-months-to-date` 1694
`altova:add-months-to-dateTime` 1692
`altova:add-seconds-to-dateTime` 1692
`altova:add-seconds-to-time` 1695
`altova:add-years-to-date` 1694
`altova:add-years-to-dateTime` 1692
`altova:age` 1704
`altova:age-details` 1704
`altova:build-date` 1701
`altova:build-duration` 1701
`altova:build-time` 1701
`altova:current-dateTime-no-TZ` 1696
`altova:current-date-no-TZ` 1696
`altova:current-time-no-TZ` 1696
`altova:date-no-TZ` 1696
`altova:dateTime-from-epoch` 1705
`altova:dateTime-from-epoch-no-TZ` 1705
`altova:dateTime-no-TZ` 1696
`altova:days-in-month` 1697
`altova:epoch-from-dateTime` 1705
`altova:hours-from-dateTimeDuration-accumulated` 1697
`altova:minutes-from-dateTimeDuration-accumulated` 1697
`altova:seconds-from-dateTimeDuration-accumulated` 1697
`altova:format-duration` 1694
`altova:parse-date` 1702
`altova:parse-dateTime` 1702

- altova:parse-duration 1694
- altova:parse-time 1702
- altova:time-no-TZ 1696
- altova:weekday-from-date 1699
- altova:weekday-from-dateTime 1699
- altova:weeknumber-from-date 1700
- altova:weeknumber-from-dateTime 1700

[ Top 1690 ]

## Add a duration to xs:dateTime XP3.1 XQ3.1

These functions add a duration to **`xs:dateTime`** and return **`xs:dateTime`**. The `xs:dateTime` type has a format of `CCYY-MM-DDThh:mm:ss.sss`. This is a concatenation of the `xs:date` and `xs:time` formats separated by the letter `T`. A timezone suffix (**`+01:00`**, for example) is optional.

### ▼ add-years-to-dateTime [altova:]

**`add-years-to-dateTime(DateTime`** *`as xs:dateTime`*, **`Years`** *`as xs:integer`***`)`** **`as xs:dateTime`** XP3.1 XQ3.1

Adds a duration in years to an `xs:dateTime` (*see examples below*). The second argument is the number of years to be added to the `xs:dateTime` supplied as the first argument. The result is of type `xs:dateTime`.

⊟ *Examples*

- **`add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` returns `2024-01-15T14:00:00`
- **`add-years-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -4)` returns `2010-01-15T14:00:00`

### ▼ add-months-to-dateTime [altova:]

**`add-months-to-dateTime(DateTime`** *`as xs:dateTime`*, **`Months`** *`as xs:integer`***`)`** **`as xs:dateTime`** XP3.1 XQ3.1

Adds a duration in months to an `xs:dateTime` (*see examples below*). The second argument is the number of months to be added to the `xs:dateTime` supplied as the first argument. The result is of type `xs:dateTime`.

⊟ *Examples*

- **`add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), 10)` returns `2014-11-15T14:00:00`
- **`add-months-to-dateTime`**`(xs:dateTime("2014-01-15T14:00:00"), -2)` returns `2013-11-15T14:00:00`

### ▼ add-days-to-dateTime [altova:]

**`add-days-to-dateTime(DateTime`** *`as xs:dateTime`*, **`Days`** *`as xs:integer`***`)`** **`as xs:dateTime`** XP3.1 XQ3.1

Adds a duration in days to an `xs:dateTime` (*see examples below*). The second argument is the number of days to be added to the `xs:dateTime` supplied as the first argument. The result is of type `xs:dateTime`.

⊟ *Examples*

- **add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), 10) returns 2014-01-25T14:00:00
- **add-days-to-dateTime**(xs:dateTime("2014-01-15T14:00:00"), -8) returns 2014-01-07T14:00:00

▼ add-hours-to-dateTime [altova:]

**add-hours-to-dateTime(DateTime** *as xs:dateTime*, **Hours** *as xs:integer*) **as xs:dateTime** **XP3.1** **XQ3.1**

Adds a duration in hours to an xs:dateTime (*see examples below*). The second argument is the number of hours to be added to the xs:dateTime supplied as the first argument. The result is of type xs:dateTime.

⊟ *Examples*

- **add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), 10) returns 2014-01-15T23:00:00
- **add-hours-to-dateTime**(xs:dateTime("2014-01-15T13:00:00"), -8) returns 2014-01-15T05:00:00

▼ add-minutes-to-dateTime [altova:]

**add-minutes-to-dateTime(DateTime** *as xs:dateTime*, **Minutes** *as xs:integer*) **as xs:dateTime** **XP3.1** **XQ3.1**

Adds a duration in minutes to an xs:dateTime (*see examples below*). The second argument is the number of minutes to be added to the xs:dateTime supplied as the first argument. The result is of type xs:dateTime.

⊟ *Examples*

- **add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), 45) returns 2014-01-15T14:55:00
- **add-minutes-to-dateTime**(xs:dateTime("2014-01-15T14:10:00"), -5) returns 2014-01-15T14:05:00

▼ add-seconds-to-dateTime [altova:]

**add-seconds-to-dateTime(DateTime** *as xs:dateTime*, **Seconds** *as xs:integer*) **as xs:dateTime** **XP3.1** **XQ3.1**

Adds a duration in seconds to an xs:dateTime (*see examples below*). The second argument is the number of seconds to be added to the xs:dateTime supplied as the first argument. The result is of type xs:dateTime.

⊟ *Examples*

- **add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), 20) returns 2014-01-15T14:00:30
- **add-seconds-to-dateTime**(xs:dateTime("2014-01-15T14:00:10"), -5) returns 2014-01-15T14:00:05

[ Top 1690 ]

## Add a duration to xs:date XP3.1 XQ3.1

These functions add a duration to **xs:date** and return **xs:date**. The `xs:date` type has a format of `CCYY-MM-DD`.

▼ add-years-to-date [altova:]

**add-years-to-date(Date** *as xs:date*, **Years** *as xs:integer*) **as xs:date** XP3.1 XQ3.1
Adds a duration in years to a date. The second argument is the number of years to be added to the `xs:date` supplied as the first argument. The result is of type `xs:date`.

⊟ *Examples*

- **add-years-to-date**(xs:date("2014-01-15"), 10) returns 2024-01-15
- **add-years-to-date**(xs:date("2014-01-15"), -4) returns 2010-01-15

▼ add-months-to-date [altova:]

**add-months-to-date(Date** *as xs:date*, **Months** *as xs:integer*) **as xs:date** XP3.1 XQ3.1
Adds a duration in months to a date. The second argument is the number of months to be added to the `xs:date` supplied as the first argument. The result is of type `xs:date`.

⊟ *Examples*

- **add-months-to-date**(xs:date("2014-01-15"), 10) returns 2014-11-15
- **add-months-to-date**(xs:date("2014-01-15"), -2) returns 2013-11-15

▼ add-days-to-date [altova:]

**add-days-to-date(Date** *as xs:date*, **Days** *as xs:integer*) **as xs:date** XP3.1 XQ3.1
Adds a duration in days to a date. The second argument is the number of days to be added to the `xs:date` supplied as the first argument. The result is of type `xs:date`.

⊟ *Examples*

- **add-days-to-date**(xs:date("2014-01-15"), 10) returns 2014-01-25
- **add-days-to-date**(xs:date("2014-01-15"), -8) returns 2014-01-07

[ Top 1690 ]

## Format and retrieve durations XP3.1 XQ3.1

These functions parse an input **xs:duration** or **xs:string** and return, respectively, an **xs:string** or **xs:duration**.

▼ format-duration [altova:]

**format-duration(Duration** *as xs:duration*, **Picture** *as xs:string*) **as xs:string** XP3.1 XQ3.1
Formats a duration, which is submitted as the first argument, according to a picture string submitted as the second argument. The output is a text string formatted according to the picture string.

⊟ *Examples*

- `format-duration(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]")` returns `"Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"`
- `format-duration(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]")` returns `"Months:03 Days:02 Hours:02 Minutes:53"`

▼ parse-duration [altova:]

**parse-duration(InputString** *as xs:string*, **Picture** *as xs:string*) **as xs:duration** XP3.1 XQ3.1

Takes a patterned string as the first argument, and a picture string as the second argument. The input string is parsed on the basis of the picture string, and an `xs:duration` is returned.

⊟ *Examples*

- `parse-duration("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]")` returns `"P2DT2H53M11.7S"`
- `parse-duration("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]")` returns `"P3M2DT2H53M"`

[ Top 1690 ]

## Add a duration to xs:time XP3.1 XQ3.1

These functions add a duration to **`xs:time`** and return **`xs:time`**. The `xs:time` type has a lexical form of `hh:mm:ss.sss`. An optional time zone may be suffixed. The letter `Z` indicates Coordinated Universal Time (UTC). All other time zones are represented by their difference from UTC in the format `+hh:mm`, or `-hh:mm`. If no time zone value is present, it is considered unknown; it is not assumed to be UTC.

▼ add-hours-to-time [altova:]

**add-hours-to-time(Time** *as xs:time*, **Hours** *as xs:integer*) **as xs:time** XP3.1 XQ3.1

Adds a duration in hours to a time. The second argument is the number of hours to be added to the `xs:time` supplied as the first argument. The result is of type `xs:time`.

⊟ *Examples*

- `add-hours-to-time(xs:time("11:00:00"), 10)` returns `21:00:00`
- `add-hours-to-time(xs:time("11:00:00"), -7)` returns `04:00:00`

▼ add-minutes-to-time [altova:]

**add-minutes-to-time(Time** *as xs:time*, **Minutes** *as xs:integer*) **as xs:time** XP3.1 XQ3.1

Adds a duration in minutes to a time. The second argument is the number of minutes to be added to the `xs:time` supplied as the first argument. The result is of type `xs:time`.

⊟ *Examples*

- `add-minutes-to-time(xs:time("14:10:00"), 45)` returns `14:55:00`
- `add-minutes-to-time(xs:time("14:10:00"), -5)` returns `14:05:00`

▼ add-seconds-to-time [altova:]

**add-seconds-to-time(Time** *as xs:time*, **Minutes** *as xs:integer*) **as xs:time** XP3.1 XQ3.1
Adds a duration in seconds to a time. The second argument is the number of seconds to be added to the `xs:time` supplied as the first argument. The result is of type `xs:time`. The Seconds component can be in the range of `0` to `59.999`.

⊟ *Examples*

- **add-seconds-to-time**`(xs:time("14:00:00"), 20)` returns `14:00:20`
- **add-seconds-to-time**`(xs:time("14:00:00"), 20.895)` returns `14:00:20.895`

[ Top 1690 ]

## Remove the timezone part from date/time datatypes XP3.1 XQ3.1

These functions remove the timezone from the current **`xs:dateTime`**, **`xs:date`**, or **`xs:time`** values, respectively. Note that the difference between `xs:dateTime` and `xs:dateTimeStamp` is that in the case of the latter the timezone part is required (while it is optional in the case of the former). So the format of an `xs:dateTimeStamp` value is: `CCYY-MM-DDThh:mm:ss.sss±hh:mm`. or `CCYY-MM-DDThh:mm:ss.sssZ`. If the date and time is read from the system clock as `xs:dateTimeStamp`, the `current-dateTime-no-TZ()` function can be used to remove the timezone if so required.

▼ current-date-no-TZ [altova:]

**current-date-no-TZ() as xs:date** XP3.1 XQ3.1
This function takes no argument. It removes the timezone part of `current-date()` (which is the current date according to the system clock) and returns an `xs:date` value.

⊟ *Examples*

If the current date is **`2014-01-15+01:00`**:

- **current-date-no-TZ**`()` returns `2014-01-15`

▼ current-dateTime-no-TZ [altova:]

**current-dateTime-no-TZ() as xs:dateTime** XP3.1 XQ3.1
This function takes no argument. It removes the timezone part of `current-dateTime()` (which is the current date-and-time according to the system clock) and returns an `xs:dateTime` value.

⊟ *Examples*

If the current dateTime is **`2014-01-15T14:00:00+01:00`**:

- **current-dateTime-no-TZ**`()` returns `2014-01-15T14:00:00`

▼ current-time-no-TZ [altova:]

**current-time-no-TZ() as xs:time** XP3.1 XQ3.1
This function takes no argument. It removes the timezone part of `current-time()` (which is the current time according to the system clock) and returns an `xs:time` value.

⊟ *Examples*

If the current time is **`14:00:00+01:00`**:

- `current-time-no-TZ()` returns `14:00:00`

▼ date-no-TZ [altova:]

**`date-no-TZ(InputDate`** *`as xs:date`***`)`** **`as xs:date`** **XP3.1** **XQ3.1**

This function takes an `xs:date` argument, removes the timezone part from it, and returns an `xs:date` value. Note that the date is not modified.

⊟ *Examples*

- `date-no-TZ(xs:date("2014-01-15+01:00"))` returns `2014-01-15`

▼ dateTime-no-TZ [altova:]

**`dateTime-no-TZ(InputDateTime`** *`as xs:dateTime`***`)`** **`as xs:dateTime`** **XP3.1** **XQ3.1**

This function takes an `xs:dateTime` argument, removes the timezone part from it, and returns an `xs:dateTime` value. Note that neither the date nor the time is modified.

⊟ *Examples*

- `dateTime-no-TZ(xs:date("2014-01-15T14:00:00+01:00"))` returns `2014-01-15T14:00:00`

▼ time-no-TZ [altova:]

**`time-no-TZ(InputTime`** *`as xs:time`***`)`** **`as xs:time`** **XP3.1** **XQ3.1**

This function takes an `xs:time` argument, removes the timezone part from it, and returns an `xs:time` value. Note that the time is not modified.

⊟ *Examples*

- `time-no-TZ(xs:time("14:00:00+01:00"))` returns `14:00:00`

[ Top 1690 ]

## Return the number of days, hours, minutes, seconds from durations XP3.1 XQ3.1

These functions return the number of days in a month, and the number of hours, minutes, and seconds, respectively, from durations.

▼ days-in-month [altova:]

**`days-in-month(Year`** *`as xs:integer`***`, Month`** *`as xs:integer`***`)`** **`as xs:integer`** **XP3.1** **XQ3.1**

Returns the number of days in the specified month. The month is specified by means of the `Year` and `Month` arguments.

⊟ *Examples*

- `days-in-month(2018, 10)` returns `31`
- `days-in-month(2018,  2)` returns `28`
- `days-in-month(2020,  2)` returns `29`

### hours-from-dayTimeDuration-accumulated

**hours-from-dayTimeDuration-accumulated(DayAndTime** *as xs:duration***) as xs:integer** XP3.1 XQ3.1

Returns the total number of hours in the duration submitted by the `DayAndTime` argument (which is of type `xs:duration`). The hours in the `Day` and `Time` components are added together to give a result that is an integer. A new hour is counted only for a full 60 minutes. Negative durations result in a negative hour value.

⊟ *Examples*

- `hours-from-dayTimeDuration-accumulated(xs:duration("P5D"))` returns `120`, which is the total number of hours in 5 days.
- `hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H"))` returns `122`, which is the total number of hours in 5 days plus 2 hours.
- `hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H60M"))` returns `123`, which is the total number of hours in 5 days plus 2 hours and 60 mins.
- `hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H119M"))` returns `123`, which is the total number of hours in 5 days plus 2 hours and 119 mins.
- `hours-from-dayTimeDuration-accumulated(xs:duration("P5DT2H120M"))` returns `124`, which is the total number of hours in 5 days plus 2 hours and 120 mins.
- `hours-from-dayTimeDuration-accumulated(xs:duration("-P5DT2H"))` returns `-122`

### minutes-from-dayTimeDuration-accumulated

**minutes-from-dayTimeDuration-accumulated(DayAndTime** *as xs:duration***) as xs:integer** XP3.1 XQ3.1

Returns the total number of minutes in the duration submitted by the `DayAndTime` argument (which is of type `xs:duration`). The minutes in the `Day` and `Time` components are added together to give a result that is an integer. Negative durations result in a negative minute value.

⊟ *Examples*

- `minutes-from-dayTimeDuration-accumulated(xs:duration("PT60M"))` returns `60`
- `minutes-from-dayTimeDuration-accumulated(xs:duration("PT1H"))` returns `60`, which is the total number of minutes in 1 hour.
- `minutes-from-dayTimeDuration-accumulated(xs:duration("PT1H40M"))` returns `100`
- `minutes-from-dayTimeDuration-accumulated(xs:duration("P1D"))` returns `1440`, which is the total number of minutes in 1 day.
- `minutes-from-dayTimeDuration-accumulated(xs:duration("-P1DT60M"))` returns `-1500`

### seconds-from-dayTimeDuration-accumulated

**seconds-from-dayTimeDuration-accumulated(DayAndTime** *as xs:duration***) as xs:integer** XP3.1 XQ3.1

Returns the total number of seconds in the duration submitted by the `DayAndTime` argument (which is of type `xs:duration`). The seconds in the `Day` and `Time` components are added together to give a result that is an integer. Negative durations result in a negative seconds value.

⊟ *Examples*

- `seconds-from-dayTimeDuration-accumulated(xs:duration("PT1M"))` returns `60`, which is the total number of seconds in 1 minute.
- `seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H"))` returns `3600`, which is the total number of seconds in 1 hour.
- `seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H2M"))` returns `3720`

- `seconds-from-dayTimeDuration-accumulated(xs:duration("P1D"))` returns `86400`, which is the total number of seconds in 1 day.
- `seconds-from-dayTimeDuration-accumulated(xs:duration("-P1DT1M"))` returns `-86460`

## Return the weekday from xs:dateTime or xs:date XP3.1 XQ3.1

These functions return the weekday (as an integer) from `xs:dateTime` or `xs:date`. The days of the week are numbered (using the American format) from `1` to `7`, with `Sunday=1`. In the European format, the week starts with Monday (=`1`). The American format, where `Sunday=1`, can be set by using the integer `0` where an integer is accepted to indicate the format.

### weekday-from-dateTime [altova:]

**`weekday-from-dateTime(DateTime`** *`as xs:dateTime`***`) as xs:integer`** **XP3.1** **XQ3.1**

Takes a date-with-time as its single argument and returns the day of the week of this date as an integer. The weekdays are numbered starting with `Sunday=1`. If the European format is required (where `Monday=1`), use the other signature of this function (*see next signature below*).

*Examples*

- `weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"))` returns `2`, which would indicate a Monday.

**`weekday-from-dateTime(DateTime`** *`as xs:dateTime`*, **`Format`** *`as xs:integer`***`) as xs:integer`** **XP3.1** **XQ3.1**

Takes a date-with-time as its first argument and returns the day of the week of this date as an integer. If the second (integer) argument is `0`, then the weekdays are numbered `1` to `7` starting with `Sunday=1`. If the second argument is an integer other than `0`, then `Monday=1`. If there is no second argument, the function is read as having the other signature of this function (*see previous signature*).

*Examples*

- `weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 1)` returns `1`, which would indicate a Monday
- `weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 4)` returns `1`, which would indicate a Monday
- `weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 0)` returns `2`, which would indicate a Monday.

### weekday-from-date [altova:]

**`weekday-from-date(Date`** *`as xs:date`***`) as xs:integer`** **XP3.1** **XQ3.1**

Takes a date as its single argument and returns the day of the week of this date as an integer. The weekdays are numbered starting with `Sunday=1`. If the European format is required (where `Monday=1`), use the other signature of this function (*see next signature below*).

*Examples*

- `weekday-from-date(xs:date("2014-02-03+01:00"))` returns `2`, which would indicate a Monday.

**`weekday-from-date(Date`** *`as xs:date`*, **`Format`** *`as xs:integer`***`) as xs:integer`** **XP3.1** **XQ3.1**

Takes a date as its first argument and returns the day of the week of this date as an integer. If the second

(`Format`) argument is `0`, then the weekdays are numbered `1` to `7` starting with `Sunday=1`. If the second argument is an integer other than `0`, then `Monday=1`. If there is no second argument, the function is read as having the other signature of this function (*see previous signature*).

⊟ *Examples*

- `weekday-from-date(xs:date("2014-02-03"), 1)` returns `1`, which would indicate a Monday
- `weekday-from-date(xs:date("2014-02-03"), 4)` returns `1`, which would indicate a Monday
- `weekday-from-date(xs:date("2014-02-03"), 0)` returns `2`, which would indicate a Monday.

[ Top 1690 ]

## Return the week number from xs:dateTime or xs:date XP2 XQ1 XP3.1 XQ3.1

These functions return the week number (as an integer) from `xs:dateTime` or `xs:date`. Week-numbering is available in the US, ISO/European, and Islamic calendar formats. Week-numbering is different in these calendar formats because the week is considered to start on different days (on Sunday in the US format, Monday in the ISO/European format, and Saturday in the Islamic format).

### weeknumber-from-date [altova:]

**`weeknumber-from-date(Date`** *`as xs:date`*, **`Calendar`** *`as xs:integer`*) **`as xs:integer`** XP2 XQ1 XP3.1 XQ3.1

Returns the week number of the submitted **`Date`** argument as an integer. The second argument (**`Calendar`**) specifies the calendar system to follow.
Supported **`Calendar`** values are:

- **`0 = US calendar`** (*week starts Sunday*)
- **`1 = ISO standard, European calendar`** (*week starts Monday*)
- **`2 = Islamic calendar`** (*week starts Saturday*)

Default is **`0`**.

⊟ *Examples*

- `weeknumber-from-date(xs:date("2014-03-23"), 0)` returns `13`
- `weeknumber-from-date(xs:date("2014-03-23"), 1)` returns `12`
- `weeknumber-from-date(xs:date("2014-03-23"), 2)` returns `13`
- `weeknumber-from-date(xs:date("2014-03-23")    )` returns `13`

The day of the date in the examples above (`2014-03-23`) is Sunday. So the US and Islamic calendars are one week ahead of the European calendar on this day.

### weeknumber-from-dateTime [altova:]

**`weeknumber-from-dateTime(DateTime`** *`as xs:dateTime`*, **`Calendar`** *`as xs:integer`*) **`as xs:integer`** XP2 XQ1 XP3.1 XQ3.1

Returns the week number of the submitted **`DateTime`** argument as an integer. The second argument (**`Calendar`**) specifies the calendar system to follow.
Supported **`Calendar`** values are:

- **`0 = US calendar`** (*week starts Sunday*)
- **`1 = ISO standard, European calendar`** (*week starts Monday*)
- **`2 = Islamic calendar`** (*week starts Saturday*)

Default is **`0`**.

⊟ *Examples*

- **`weeknumber-from-dateTime`**`(xs:dateTime("2014-03-23T00:00:00"), 0)` returns `13`
- **`weeknumber-from-dateTime`**`(xs:dateTime("2014-03-23T00:00:00"), 1)` returns `12`
- **`weeknumber-from-dateTime`**`(xs:dateTime("2014-03-23T00:00:00"), 2)` returns `13`
- **`weeknumber-from-dateTime`**`(xs:dateTime("2014-03-23T00:00:00")    )` returns `13`

The day of the dateTime in the examples above (`2014-03-23T00:00:00`) is Sunday. So the US and Islamic calendars are one week ahead of the European calendar on this day.

[ Top 1690 ]

## Build date, time, and duration datatypes from their lexical components XP3.1 XQ3.1

The functions take the lexical components of the `xs:date`, `xs:time`, or `xs:duration` datatype as input arguments and combine them to build the respective datatype.

### build-date [altova:]

**`build-date(Year`** *`as xs:integer`*`,` **`Month`** *`as xs:integer`*`,` **`Date`** *`as xs:integer`***`)`** **`as xs:date`** XP3.1 XQ3.1

The first, second, and third arguments are, respectively, the year, month, and date. They are combined to build a value of `xs:date` type. The values of the integers must be within the correct range of that particular date part. For example, the second argument (for the month part) should not be greater than `12`.

⊟ *Examples*

- **`build-date`**`(2014, 2, 03)` returns `2014-02-03`

### build-time [altova:]

**`build-time(Hours`** *`as xs:integer`*`,` **`Minutes`** *`as xs:integer`*`,` **`Seconds`** *`as xs:integer`***`)`** **`as xs:time`** XP3.1 XQ3.1

The first, second, and third arguments are, respectively, the hour (`0` to `23`), minutes (`0` to `59`), and seconds (`0` to `59`) values. They are combined to build a value of `xs:time` type. The values of the integers must be within the correct range of that particular time part. For example, the second (`Minutes`) argument should not be greater than `59`. To add a timezone part to the value, use the other signature of this function (*see next signature*).

⊟ *Examples*

- **`build-time`**`(23, 4, 57)` returns `23:04:57`

**`build-time(Hours`** *`as xs:integer`*`,` **`Minutes`** *`as xs:integer`*`,` **`Seconds`** *`as xs:integer`*`,` **`TimeZone`** *`as xs:string`***`)`** **`as xs:time`** XP3.1 XQ3.1

The first, second, and third arguments are, respectively, the hour (`0` to `23`), minutes (`0` to `59`), and seconds

(0 to 59) values. The fourth argument is a string that provides the timezone part of the value. The four arguments are combined to build a value of `xs:time` type. The values of the integers must be within the correct range of that particular time part. For example, the second (`Minutes`) argument should not be greater than 59.

⊟ *Examples*

- `build-time(23, 4, 57, '+1')` returns `23:04:57+01:00`

▼ build-duration [altova:]

**`build-duration(Years`** *`as xs:integer`*, **`Months`** *`as xs:integer`***`)`** **`as xs:yearMonthDuration`** **XP3.1** **XQ3.1**

Takes two arguments to build a value of type `xs:yearMonthDuration`. The first argument provides the `Years` part of the duration value, while the second argument provides the `Months` part. If the second (`Months`) argument is greater than or equal to 12, then the integer is divided by 12; the quotient is added to the first argument to provide the `Years` part of the duration value while the remainder (of the division) provides the `Months` part. To build a duration of type `xs:dayTimeDuration`., see the next signature.

⊟ *Examples*

- `build-duration(2, 10)` returns `P2Y10M`
- `build-duration(14, 27)` returns `P16Y3M`
- `build-duration(2, 24)` returns `P4Y`

**`build-duration(Days`** *`as xs:integer`*, **`Hours`** *`as xs:integer`*, **`Minutes`** *`as xs:integer`*, **`Seconds`** *`as xs:integer`***`)`** as **`xs:dayTimeDuration`** **XP3.1** **XQ3.1**

Takes four arguments and combines them to build a value of type `xs:dayTimeDuration`. The first argument provides the `Days` part of the duration value, the second, third, and fourth arguments provide, respectively, the `Hours`, `Minutes`, and `Seconds` parts of the duration value. Each of the three Time arguments is converted to an equivalent value in terms of the next higher unit and the result is used for calculation of the total duration value. For example, 72 seconds is converted to `1M+12S` (1 minute and 12 seconds), and this value is used for calculation of the total duration value. To build a duration of type `xs:yearMonthDuration`., see the previous signature.

⊟ *Examples*

- `build-duration(2, 10, 3, 56)` returns `P2DT10H3M56S`
- `build-duration(1, 0, 100, 0)` returns `P1DT1H40M`
- `build-duration(1, 0, 0, 3600)` returns `P1DT1H`

[ Top 1690 ]

## Construct date, dateTime, and time datatypes from string input XP2 XQ1 XP3.1 XQ3.1

These functions take strings as arguments and construct `xs:date`, `xs:dateTime`, or `xs:time` datatypes. The string is analyzed for components of the datatype based on a submitted pattern argument.

▼ parse-date [altova:]

**`parse-date(Date`** *`as xs:string`*, **`DatePattern`** *`as xs:string`***`)`** **`as xs:date`** **XP2** **XQ1** **XP3.1** **XQ3.1**

Returns the input string **`Date`** as an **`xs:date`** value. The second argument **`DatePattern`** specifies the pattern (sequence of components) of the input string. **`DatePattern`** is described with the component specifiers listed below and with component separators that can be any character. See the examples below.

| | |
|---|---|
| **D** | Date |
| **M** | Month |
| **Y** | Year |

The pattern in **`DatePattern`** must match the pattern in **`Date`**. Since the output is of type **`xs:date`**, the output will always have the lexical format **`YYYY-MM-DD`**.

⊟ *Examples*

- `parse-date(xs:string("09-12-2014"), "[D]-[M]-[Y]")` returns `2014-12-09`
- `parse-date(xs:string("09-12-2014"), "[M]-[D]-[Y]")` returns `2014-09-12`
- `parse-date("06/03/2014", "[M]/[D]/[Y]")` returns `2014-06-03`
- `parse-date("06 03 2014", "[M] [D] [Y]")` returns `2014-06-03`
- `parse-date("6 3 2014", "[M] [D] [Y]")` returns `2014-06-03`

### parse-dateTime [altova:]

**`parse-dateTime(DateTime`** *`as xs:string`*, **`DateTimePattern`** *`as xs:string`*) **`as xs:dateTime`** XP2 XQ1 XP3.1 XQ3.1

Returns the input string **`DateTime`** as an **`xs:dateTime`** value.The second argument **`DateTimePattern`** specifies the pattern (sequence of components) of the input string. **`DateTimePattern`** is described with the component specifiers listed below and with component separators that can be any character. See the examples below.

| | |
|---|---|
| **D** | Date |
| **M** | Month |
| **Y** | Year |
| **H** | Hour |
| **m** | minutes |
| **s** | seconds |

The pattern in **`DateTimePattern`** must match the pattern in **`DateTime`**. Since the output is of type **`xs:dateTime`**, the output will always have the lexical format **`YYYY-MM-DDTHH:mm:ss`**.

⊟ *Examples*

- `parse-dateTime(xs:string("09-12-2014 13:56:24"), "[M]-[D]-[Y] [H]:[m]:[s]")` returns `2014-09-12T13:56:24`
- `parse-dateTime("time=13:56:24; date=09-12-2014", "time=[H]:[m]:[s]; date=[D]-[M]-[Y]")` returns `2014-12-09T13:56:24`

### parse-time [altova:]

**`parse-time(Time`** *`as xs:string`*, **`TimePattern`** *`as xs:string`*) **`as xs:time`** XP2 XQ1 XP3.1 XQ3.1

Returns the input string **`Time`** as an **`xs:time`** value.The second argument **`TimePattern`** specifies the pattern (sequence of components) of the input string. **`TimePattern`** is described with the component specifiers listed below and with component separators that can be any character. See the examples below.

| | |
|---|---|
| **H** | Hour |
| **m** | minutes |
| **s** | seconds |

The pattern in **`TimePattern`** must match the pattern in **`Time`**. Since the output is of type **`xs:time`**, the output will always have the lexical format **`HH:mm:ss`**.

⊟ *Examples*

- `parse-time(xs:string("13:56:24"), "[H]:[m]:[s]")` returns `13:56:24`
- `parse-time("13-56-24", "[H]-[m]")` returns `13:56:00`
- `parse-time("time=13h56m24s", "time=[H]h[m]m[s]s")` returns `13:56:24`
- `parse-time("time=24s56m13h", "time=[s]s[m]m[H]h")` returns `13:56:24`

[ Top 1690 ]

## Age-related functions XP3.1 XQ3.1

These functions return the age as calculated (i) between one input argument date and the current date, or (ii) between two input argument dates. The **`altova:age`** function returns the age in terms of years, the **`altova:age-details`** function returns the age as a sequence of three integers giving the years, months, and days of the age.

### age [altova:]

**`age(StartDate`** *`as xs:date`***`) as xs:integer`** XP3.1 XQ3.1

Returns an integer that is the age *in years* of some object, counting from a start-date submitted as the argument and ending with the current date (taken from the system clock). If the input argument is a date anything greater than or equal to one year in the future, the return value will be negative.

⊟ *Examples*

If the current date is **`2014-01-15`**:

- `age(xs:date("2013-01-15"))` returns `1`
- `age(xs:date("2013-01-16"))` returns `0`
- `age(xs:date("2015-01-15"))` returns `-1`
- `age(xs:date("2015-01-14"))` returns `0`

**`age(StartDate`** *`as xs:date`***`, EndDate`** *`as xs:date`***`) as xs:integer`** XP3.1 XQ3.1

Returns an integer that is the age *in years* of some object, counting from a start-date that is submitted as the first argument up to an end-date that is the second argument. The return value will be negative if the first argument is one year or more later than the second argument.

⊟ *Examples*

If the current date is **`2014-01-15`**:

- `age(xs:date("2000-01-15"), xs:date("2010-01-15"))` returns `10`
- `age(xs:date("2000-01-15"), current-date())` returns `14` if the current date is `2014-01-15`
- `age(xs:date("2014-01-15"), xs:date("2010-01-15"))` returns `-4`

▼ age-details [altova:]

**age-details(InputDate** *as xs:date***)** as (**xs:integer**)* **XP3.1** **XQ3.1**

Returns three integers that are, respectively, the years, months, and days between the date that is submitted as the argument and the current date (taken from the system clock). The sum of the returned `years+months+days` together gives the total time difference between the two dates (the input date and the current date). The input date may have a value earlier or later than the current date, but whether the input date is earlier or later is not indicated by the sign of the return values; the return values are always positive.

⊟ *Examples*

If the current date is **`2014-01-15`**:

- **`age-details`**`(xs:date("2014-01-16"))` returns `(0 0 1)`
- **`age-details`**`(xs:date("2014-01-14"))` returns `(0 0 1)`
- **`age-details`**`(xs:date("2013-01-16"))` returns `(1 0 1)`
- **`age-details`**`(current-date())` returns `(0 0 0)`

**age-details(Date-1** *as xs:date*, **Date-2** *as xs:date***)** as (**xs:integer**)* **XP3.1** **XQ3.1**

Returns three integers that are, respectively, the years, months, and days between the two argument dates. The sum of the returned `years+months+days` together gives the total time difference between the two input dates; it does not matter whether the earlier or later of the two dates is submitted as the first argument. The return values do not indicate whether the input date occurs earlier or later than the current date. Return values are always positive.

⊟ *Examples*

- **`age-details`**`(xs:date("2014-01-16"), xs:date("2014-01-15"))` returns `(0 0 1)`
- **`age-details`**`(xs:date("2014-01-15"), xs:date("2014-01-16"))` returns `(0 0 1)`

[ **Top** 1690 ]

## Epoch time (Unix time) functions **XP3.1** **XQ3.1**

Epoch time is a time system used on Unix systems. It defines any given point in time as being the number of seconds that have elapsed since 00:00:00 UTC on 1 January 1970. Altova's Epoch time extension functions convert **`xs:dateTime`** values to Epoch time values and vice versa.

▼ dateTime-from-epoch [altova:]

**dateTime-from-epoch(Epoch** *as xs:decimal* **as xs:dateTime** **XP3.1** **XQ3.1**

Epoch time is a time system used on Unix systems. It defines any given point in time as being the number of seconds that have elapsed since 00:00:00 UTC on 1 January 1970. The **`dateTime-from-epoch`** function returns the **`xs:dateTime`** equivalent of an Epoch time, adjusts it for the local timezone, and includes the timezone information in the result.

The function takes an `xs:decimal` argument and returns an `xs:dateTime` value that includes a **`TZ`** (timezone) part. The result is obtained by calculating the UTC **`dateTime`** equivalet of the Epoch time, and adding to it the local timezone (taken from the system clock). For example, if the function is executed on a machine that has been set to be in a timezone of +01:00 (relative to UTC), then after the UTC **`dateTime`** equivalent has been calculated, one hour will be added to the result. The timezone information, which is an optional lexical part of the **xs:dateTime** result, is also reported in the **dateTime** result. Compare this result with that of **dateTime-from-epoch-no-TZ**, and also see the function **epoch-from-dateTime**.

⊟ *Examples*

The examples below assume a local timezone of UTC +01:00. Consequently, the UTC **dateTime** equivalent of the submitted Epoch time will be incremented by one hour. The timezone is reported in the result.

- `dateTime-from-epoch(34)` returns `1970-01-01T01:00:34+01:00`
- `dateTime-from-epoch(62)` returns `1970-01-01T01:01:02+01:00`

### ▼ dateTime-from-epoch-no-TZ [altova:]

**dateTime-from-epoch-no-TZ(Epoch** *as xs:decimal* **as xs:dateTime** **XP3.1** **XQ3.1**

Epoch time is a time system used on Unix systems. It defines any given point in time as being the number of seconds that have elapsed since 00:00:00 UTC on 1 January 1970. The **dateTime-from-epoch-no-TZ** function returns the **xs:dateTime** equivalent of an Epoch time, adjusts it for the local timezone, but does not include the timezone information in the result.

The function takes an `xs:decimal` argument and returns an `xs:dateTime` value that does not includes a **TZ** (timezone) part. The result is obtained by calculating the UTC **dateTime** equivalet of the Epoch time, and adding to it the local timezone (taken from the system clock). For example, if the function is executed on a machine that has been set to be in a timezone of +01:00 (relative to UTC), then after the UTC **dateTime** equivalent has been calculated, one hour will be added to the result. The timezone information, which is an optional lexical part of the **xs:dateTime** result, is not reported in the **dateTime** result. Compare this result with that of **dateTime-from-epoch**, and also see the function **epoch-from-dateTime**.

⊟ *Examples*

The examples below assume a local timezone of UTC +01:00. Consequently, the UTC **dateTime** equivalent of the submitted Epoch time will be incremented by one hour. The timezone is not reported in the result.

- `dateTime-from-epoch(34)` returns `1970-01-01T01:00:34`
- `dateTime-from-epoch(62)` returns `1970-01-01T01:01:02`

### ▼ epoch-from-dateTime [altova:]

**epoch-from-dateTime(dateTimeValue** *as xs:dateTime***)** **as xs:decimal** **XP3.1** **XQ3.1**

Epoch time is a time system used on Unix systems. It defines any given point in time as being the number of seconds that have elapsed since 00:00:00 UTC on 1 January 1970. The **epoch-from-dateTime** function returns the Epoch time equivalent of the **xs:dateTime** that is submitted as the argument of the function. Note that you might have to explicitly construct the **xs:dateTime** value. The submitted **xs:dateTime** value may or may not contain the optional **TZ** (timezone) part.

Whether the timezone part is submitted as part of the argument or not, the local timezone offset (taken from the system clock) is subtracted from the submitted **dateTimeValue** argument. This produces the equivalent UTC time, from which the equivalent Epoch time is calculated. For example, if the function is executed on a machine that has been set to be in a timezone of +01:00 (relative to UTC), then one hour is subtracted from the submitted **`dateTimeValue`** before the Epoch value is calculated. Also see the function **`dateTime-from-epoch`**.

⊟ *Examples*

The examples below assume a local timezone of UTC +01:00. Consequently, one hour will be subtracted from the submitted **`dateTime`** before the Epoch time is calculated.

- **`epoch-from-dateTime`**`(xs:dateTime("1970-01-01T01:00:34+01:00"))` returns `34`
- **`epoch-from-dateTime`**`(xs:dateTime("1970-01-01T01:00:34"))` returns `34`
- **`epoch-from-dateTime`**`(xs:dateTime("2021-04-01T11:22:33"))` returns `1617272553`

[ Top[1690] ]

### 30.1.1.2 XPath/XQuery Functions: Geolocation

The following geolocation XPath/XQuery extension functions are supported in the current version of MobileTogether Designer.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | XP1 XP2 XP3.1 |
| *XSLT functions (used in XPath expressions in XSLT):* | XSLT1 XSLT2 XSLT3 |
| *XQuery functions (used in XQuery expressions in XQuery):* | XQ1 XQ3.1 |

▼ format-geolocation [altova:]

**`format-geolocation(Latitude`** *`as xs:decimal`*, **`Longitude`** *`as xs:decimal`*, **`GeolocationOutputStringFormat`** *`as xs:integer`*) **`as xs:string`** XP3.1 XQ3.1

Takes the latitude and longitude as the first two arguments, and outputs the geolocation as a string. The third argument, **`GeolocationOutputStringFormat`**, is the format of the geolocation output string; it uses integer values from `1` to `4` to identify the output string format (*see 'Geolocation output string formats' below*). Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** The `image-exif-data`[1718] function and the Exif metadata's attributes can be used to supply the input strings.

⊟ *Examples*

- `format-geolocation(33.33, -22.22, 4)` returns the `xs:string` `"33.33 -22.22"`
- `format-geolocation(33.33, -22.22, 2)` returns the `xs:string` `"33.33N 22.22W"`
- `format-geolocation(-33.33, 22.22, 2)` returns the `xs:string` `"33.33S 22.22E"`
- `format-geolocation(33.33, -22.22, 1)` returns the `xs:string` `"33°19'48.00"S 22° 13'12.00"E"`

### *Geolocation output string formats:*

The supplied latitude and longitude is formatted in one of the output formats given below. The desired format is identified by its integer ID (1 to 4). Latitude values range from +90 to -90 (N to S). Longitude values range from +180 to -180 (E to W).

| **1** |
|---|
| Degrees, minutes, decimal seconds, with suffixed orientation (`N/S, E/W`)<br>`D°M'S.SS"N/S  D°M'S.SS"E/W`<br>*Example*: **`33°55'11.11"N  22°44'66.66"W`** |

| **2** |
|---|
| Decimal degrees, with suffixed orientation (`N/S`, `E/W`)<br>`D.DDN/S  D.DDE/W`<br>*Example*: **`33.33N  22.22W`** |

| **3** |
|---|
| Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); plus sign for (**`N/E`**) is optional<br>`+/-D°M'S.SS"  +/-D°M'S.SS"`<br>*Example*: **`33°55'11.11"  -22°44'66.66"`** |

| **4** |
|---|
| Decimal degrees, with prefixed sign (`+/-`); plus sign for (**`N/E`**) is optional<br>`+/-D.DD  +/-D.DD`<br>*Example*: **`33.33 -22.22`** |

### *Altova Exif Attribute: Geolocation*

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151° 13'11.73"E |

### parse-geolocation [altova:]

**`parse-geolocation(GeolocationInputString`** *`as xs:string`***`)`** **`as xs:decimal+`** **`XP3.1`** **`XQ3.1`**

Parses the supplied `GeolocationInputString` argument and returns the geolocation's latitude and longitude (in that order) as a sequence two `xs:decimal` items. The formats in which the geolocation input string can be supplied are listed below.

**Note:** The `image-exif-data`[1718] function and the Exif metadata's `@Geolocation`[1718] attribute can be used to supply the geolocation input string (*see example below*).

*Examples*

- `parse-geolocation("33.33  -22.22")` returns the sequence of two `xs:decimals (33.33, 22.22)`
- `parse-geolocation("48°51'29.6""N  24°17'40.2""")` returns the sequence of two `xs:decimals (48.8582222222222, 24.2945)`
- `parse-geolocation('48°51''29.6"N  24°17''40.2"')` returns the sequence of two `xs:decimals (48.8582222222222, 24.2945)`
- `parse-geolocation( image-exif-data(//MyImages/Image20141130.01)/@Geolocation )` returns a sequence of two `xs:decimals`

*Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (`"`) while unit indicators that are escaped are highlighted in blue (`""`).

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example*: **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example*: **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example*: **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example*: **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example:* **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example:* **`33.33  -22.22`**

*Examples of format-combinations:*
**`33.33N  -22°44'55.25"`**
**`33.33  22°44'55.25"W`**
**`33.33  22.45`**

*Altova Exif Attribute: Geolocation*

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151° 13'11.73"E |

geolocation-distance-km [altova:]

**`geolocation-distance-km(GeolocationInputString-1`** *`as xs:string`*, **`GeolocationInputString-2`** *`as xs:string`*) **`as xs:decimal`** **XP3.1** **XQ3.1**

Calculates the distance between two geolocations in kilometers. The formats in which the geolocation input string can be supplied are listed below. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** The `image-exif-data`[1718] function and the Exif metadata's `@Geolocation`[1718] attribute can be used to supply geolocation input strings.

*Examples*

- `geolocation-distance-km("33.33  -22.22", "48°51'29.6""N  24°17'40.2""")` returns the `xs:decimal 4183.08132372392`

*Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (`"`) while unit indicators that are escaped are highlighted in blue (`""`).

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example:* **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example:* **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example:* **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example:* **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example:* **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example:* **`33.33  -22.22`**

*Examples of format-combinations:*

**`33.33N  -22°44'55.25"`**
**`33.33  22°44'55.25"W`**
**`33.33  22.45`**

⊟ *Altova Exif Attribute: Geolocation*

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151°13'11.73"E |

▼ geolocation-distance-mi [altova:]

**geolocation-distance-mi(GeolocationInputString-1** *as xs:string*, **GeolocationInputString-2** *as xs:string*) **as xs:decimal** XP3.1 XQ3.1
Calculates the distance between two geolocations in miles. The formats in which a geolocation input string can be supplied are listed below. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** The `image-exif-data` 1718 function and the Exif metadata's `@Geolocation` 1718 attribute can be used to supply geolocation input strings.

⊟ *Examples*

- `geolocation-distance-mi("33.33  -22.22", "48°51'29.6""N  24°17'40.2""")` returns the `xs:decimal 2599.40652340653`

⊟ *Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (`"`) while unit indicators that are escaped are highlighted in blue (`""`).

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example*: **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example*: **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example*: **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example*: **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example*: **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example*: **`33.33  -22.22`**

*Examples of format-combinations:*

```
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45
```

⊟ *Altova Exif Attribute: Geolocation*

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151°13'11.73"E |

geolocations-bounding-rectangle [altova:]

**`geolocations-bounding-rectangle(Geolocations`** *`as xs:sequence`*, **`GeolocationOutputStringFormat`** *`as xs:integer`***`)`** **`as xs:string`** **XP3.1** **XQ3.1**

Takes a sequence of strings as its first argument; each string in the sequence is a geolocation. The function returns a sequence of two strings which are, respectively, the top-left and bottom-right geolocation coordinates of a bounding rectangle that is optimally sized to enclose all the geolocations submitted in the first argument. The formats in which a geolocation input string can be supplied are listed below (*see 'Geolocation input string formats'*). Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

The function's second argument specifies the format of the two geolocation strings in the output sequence. The argument takes an integer value from `1` to `4`, where each value identifies a different geolocation string format (*see 'Geolocation output string formats' below*).

**Note:** The `image-exif-data`[1718] function and the Exif metadata's attributes can be used to supply the input strings.

⊟ *Examples*

- **`geolocations-bounding-rectangle`**`(("48.2143531 16.3707266", "51.50939 -0.11832"), 1)` returns the sequence `("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°22'14.61576"E")`
- **`geolocations-bounding-rectangle`**`(("48.2143531 16.3707266", "51.50939 -0.11832", "42.5584577 -70.8893334"), 4)` returns the sequence `("51.50939 -70.8893334", "42.5584577 16.3707266")`

⊟ *Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`).

Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (") while unit indicators that are escaped are highlighted in blue ("").

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example:* **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example:* **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example:* **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example:* **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example:* **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example:* **`33.33  -22.22`**

*Examples of format-combinations:*
**`33.33N  -22°44'55.25"`**
**`33.33  22°44'55.25"W`**
**`33.33  22.45`**

⊟ *Geolocation output string formats:*

The supplied latitude and longitude is formatted in one of the output formats given below. The desired format is identified by its integer ID (1 to 4). Latitude values range from +90 to -90 (N to S). Longitude values range from +180 to -180 (E to W).

| **1** |
|---|
| Degrees, minutes, decimal seconds, with suffixed orientation (`N/S, E/W`)<br>`D°M'S.SS"N/S  D°M'S.SS"E/W`<br>*Example:* **`33°55'11.11"N  22°44'66.66"W`** |

| **2** |
|---|

| |
|---|
| Decimal degrees, with suffixed orientation (`N/S`, `E/W`)<br>`D.DDN/S  D.DDE/W`<br>*Example*: **`33.33N  22.22W`** |

| **3** |
|---|
| Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); plus sign for (**`N/E`**) is optional<br>`+/-D°M'S.SS"  +/-D°M'S.SS"`<br>*Example*: **`33°55'11.11"  -22°44'66.66"`** |

| **4** |
|---|
| Decimal degrees, with prefixed sign (`+/-`); plus sign for (**`N/E`**) is optional<br>`+/-D.DD  +/-D.DD`<br>*Example*: **`33.33 -22.22`** |

⊟ *Altova Exif Attribute: Geolocation*

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151°13'11.73"E |

▼ geolocation-within-polygon [altova:]

**`geolocation-within-polygon(Geolocation`** *`as xs:string`*, **`((PolygonPoint`** *`as xs:string`***`)+))`** **`as xs:boolean`** **XP3.1** **XQ3.1**

Determines whether **`Geolocation`** (the first argument) is within the polygonal area described by the **`PolygonPoint`** arguments. If the `PolygonPoint` arguments do not form a closed figure (formed when the first point and the last point are the same), then the first point is implicitly added as the last point in order to close the figure. All the arguments (`Geolocation` and `PolygonPoint+`) are given by geolocation input strings (*formats listed below*). If the `Geolocation` argument is within the polygonal area, then the function returns `true()`; otherwise it returns `false()`. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** The `image-exif-data`[1718] function and the Exif metadata's `@Geolocation`[1718] attribute can be used to supply geolocation input strings.

⊟ *Examples*

- **`geolocation-within-polygon(`**`"33 -22", ("58 -32", "-78 -55", "48 24", "58 -32")`**`)`** returns `true()`

- `geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48 24"))` returns `true()`
- `geolocation-within-polygon("33 -22", ("58 -32", "-78 -55", "48°51'29.6""N 24°17'40.2"""))` returns `true()`

*Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (`"`) while unit indicators that are escaped are highlighted in blue (`""`).

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example:* **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example:* **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example:* **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example:* **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example:* **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example:* **`33.33  -22.22`**

*Examples of format-combinations:*
**`33.33N  -22°44'55.25"`**
**`33.33  22°44'55.25"W`**
**`33.33  22.45`**

*Altova Exif Attribute: Geolocation*

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151° 13'11.73"E |

▼ geolocation-within-rectangle [altova:]

**`geolocation-within-rectangle(Geolocation`** *`as xs:string`*, **`RectCorner-1`** *`as xs:string`*, **`RectCorner-2`** *`as xs:string`*) **`as xs:boolean`** **XP3.1** **XQ3.1**

Determines whether **`Geolocation`** (the first argument) is within the rectangle defined by the second and third arguments, **`RectCorner-1`** and **`RectCorner-2`**, which specify opposite corners of the rectangle. All the arguments (`Geolocation`, **`RectCorner-1`** and **`RectCorner-2`**) are given by geolocation input strings (*formats listed below*). If the `Geolocation` argument is within the rectangle, then the function returns `true()`; otherwise it returns `false()`. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** The `image-exif-data` 1718 function and the Exif metadata's `@Geolocation` 1718 attribute can be used to supply geolocation input strings.

⊟ *Examples*

- **`geolocation-within-rectangle(`**`"33 -22", "58 -32", "-48 24"`) returns `true()`
- **`geolocation-within-rectangle(`**`"33 -22", "58 -32", "48 24"`) returns `false()`
- **`geolocation-within-rectangle(`**`"33 -22", "58 -32", "48°51'29.6""S  24°17'40.2"""`) returns `true()`

⊟ *Geolocation input string formats:*

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from `+90` to `-90` (`N` to `S`). Longitude values range from `+180` to `-180` (`E` to `W`).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (`"`) while unit indicators that are escaped are highlighted in blue (`""`).

- Degrees, minutes, decimal seconds, with suffixed orientation (`N/S`, `E/W`)
  `D°M'S.SS"N/S  D°M'S.SS"W/E`
  *Example*: **`33°55'11.11"N  22°44'55.25"W`**

- Degrees, minutes, decimal seconds, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M'S.SS"  +/-D°M'S.SS"`
  *Example:* **`33°55'11.11"  -22°44'55.25"`**

- Degrees, decimal minutes, with suffixed orientation (`N/S`, `E/W`)
  `D°M.MM'N/S  D°M.MM'W/E`
  *Example:* **`33°55.55'N  22°44.44'W`**

- Degrees, decimal minutes, with prefixed sign (`+/-`); the plus sign for (**`N/E`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  *Example:* **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `E/W`)
  `D.DDN/S  D.DDW/E`
  *Example:* **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**`N/S E/W`**) is optional
  `+/-D.DD  +/-D.DD`
  *Example:* **`33.33  -22.22`**

*Examples of format-combinations:*
**`33.33N  -22°44'55.25"`**
**`33.33  22°44'55.25"W`**
**`33.33  22.45`**

*Altova Exif Attribute: Geolocation*

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151°13'11.73"E |

[ **Top** 1707 ]

## 30.1.1.3 XPath/XQuery Functions: Image-Related

The following image-related XPath/XQuery extension functions are supported in the current version of MobileTogether Designer.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | **XP1** **XP2** **XP3.1** |
| *XSLT functions (used in XPath expressions in XSLT):* | **XSLT1** **XSLT2** **XSLT3** |
| *XQuery functions (used in XQuery expressions in XQuery):* | **XQ1** **XQ3.1** |

▼ suggested-image-file-extension [altova:]

**`suggested-image-file-extension(Base64String`** *`as string`***`) as string?`** **XP3.1** **XQ3.1**

Takes the Base64 encoding of an image file as its argument and returns the file extension of the image as recorded in the Base64-encoding of the image. The returned value is a suggestion based on the image type information available in the encoding. If this information is not available, then an empty string is returned. This function is useful if you wish to save a Base64 image as a file and wish to dynamically retrieve an appropriate file extension.

⊟ *Examples*

- **`suggested-image-file-extension`**`(/MyImages/MobilePhone/Image20141130.01)` returns `'jpg'`
- **`suggested-image-file-extension`**`($XML1/Staff/Person/@photo)` returns `''`

In the examples above, the nodes supplied as the argument of the function are assumed to contain a Base64-encoded image. The first example retrieves `jpg` as the file's type and extension. In the second example, the submitted Base64 encoding does not provide usable file extension information.

▼ mt-transform-image [altova:]

**`mt-transform-image(Base64Image`** *`as Base64BinaryString`***`, Size`** *`as item()+`***`, Rotation`** *`as xs:integer`***`, Quality`** *`as xs:integer`***`) as Base64BinaryString`** **XP3.1** **XQ3.1**

Takes a Base64-encoded image as its first argument and returns a transformed Base64-encoded image. The second, third, and fourth arguments are the image parameters that are transformed: size, rotation, and quality.

- The **`Size`** argument provides three resizing options.

| | |
|---|---|
| `(X,Y)` | Absolute pixel values. The aspect ratio is not maintained. The order of height and width does not matter since height and width are automatically selected according to the long and short sides of the image. The value is entered as a sequence of two integer items; the parentheses are required. |
| `X` | Proportionally resizes the image with `X` as the new longer side in pixels; aspect ratio is maintained. The value is an integer, and is entered without quotes. |

| | |
|---|---|
| `'X%'` | Resizes the image to the given percentage of the original dimensions. The value must be entered as a string, in quotes. |

- **`Rotation`** can be one of the following values: `90`, `180`, `270`, `-90`, `-180`, `-270`. These are rotation values in degrees of a circle. Positive values rotate the image clockwise; negative values rotate the image counter-clockwise. Note that you can use the Altova Exif attribute **`OrientationDegree`** to obtain the current rotation of the image in degrees (`0`, `90`, `180`, `270`) from the Exif **`Orientation`** tag of the image. However, since the **`OrientationDegree`** attribute is obtained from the **`Orientation`** tag of the Exif data, it will be available only if the **`Orientation`** tag is present in the Exif data (*see the description of OrientationDegree below*).
- **`Quality`** can be any value between `0` and `100` and refers to values on the `IJG` quality scale for JPEG compression; it is not a percentage indicator of quality. The tradeoff is between file size and quality. For a full-color source image, `75` is generally considered an optimal value. If `75` produces unsatisfactory results, increase the value.

**Note:** If Exif data is present in the original image, it will be removed during the transformation, and the transformed image will not contain Exif data.

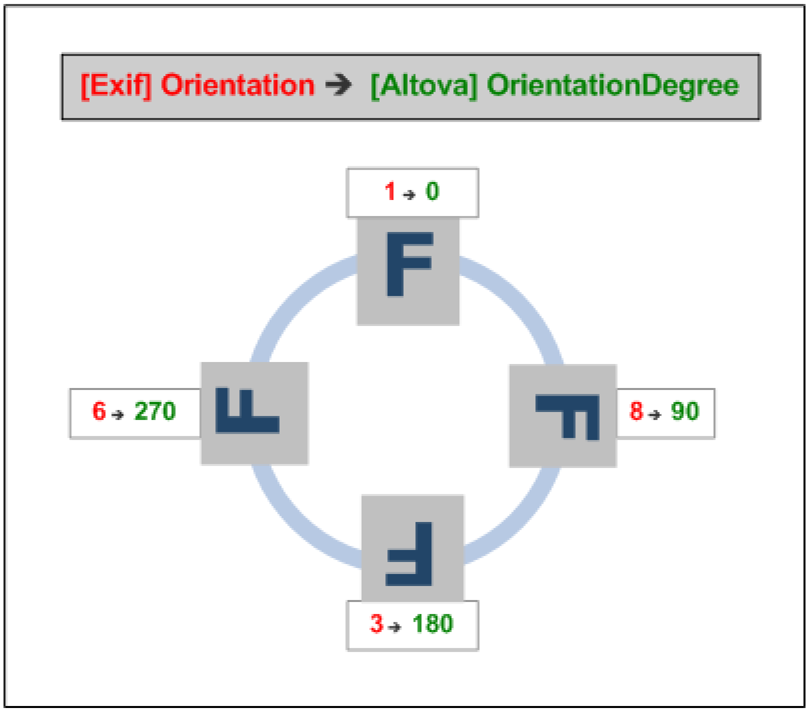
⊟ *Examples*

- `mt-transform-image(Images/Image[@id='43'], '50%', 90, 75)`
  The function takes as its input an image that is stored as a Base64-encoded string in the descendant `Images/Image` node that has an `@id` value of `43`. The function returns a transformed image. The transformed image is resized to 50%, rotated 90 degrees clockwise, and given a quality level of 75.
- `mt-transform-image(Images/Image[@id='43'], 400, 90, 75)`
  The function produces the same result as the previous example, except that the long side is set to a specific value of `400` pixels; the aspect ratio of the original image is maintained.
- `mt-transform-image(Images/Image[@id='43'], (400, 280), image-exif-data($XML1/$XML1/Images/ReferenceImage)/@OrientationDegree, 75)`
  This example selects the same image as in the previous examples, and sets the same quality value (`75`). The image size is set at `400x280` pixels, and the `Rotation` value is obtained from the **`@OrientationDegree`** attribute of a Base64-encoded image in the `ReferenceImage` node.

⊟ *Altova Exif Attribute: OrientationDegree*

The Altova XPath/XQuery Engine generates the custom attribute **`OrientationDegree`** from the Exif metadata tag **`Orientation`**.

**`OrientationDegree`** translates the standard Exif tag **`Orientation`** from an integer value (**`1`**, **`8`**, **`3`**, or **`6`**) to the respective degree values of each (**`0`**, **`90`**, **`180`**, **`270`**), as shown in the figure below. Note that there are no translations of the **`Orientation`** values of **`2`**, **`4`**, **`5`**, **`7`**. (These orientations are obtained by flipping image 1 across its vertical center axis to get the image with a value of 2, and then rotating this image in 90-degree jumps clockwise to get the values of 7, 4, and 5, respectively).



⊟ *Listing of standard Exif meta tags*

- `ImageWidth`
- `ImageLength`
- `BitsPerSample`
- `Compression`
- `PhotometricInterpretation`
- `Orientation`
- `SamplesPerPixel`
- `PlanarConfiguration`
- `YCbCrSubSampling`
- `YCbCrPositioning`
- `XResolution`
- `YResolution`
- `ResolutionUnit`
- `StripOffsets`
- `RowsPerStrip`
- `StripByteCounts`
- `JPEGInterchangeFormat`
- `JPEGInterchangeFormatLength`
- `TransferFunction`
- `WhitePoint`
- `PrimaryChromaticities`
- `YCbCrCoefficients`
- `ReferenceBlackWhite`
- `DateTime`
- `ImageDescription`
- `Make`

- `Model`
- `Software`
- `Artist`
- `Copyright`

`------------------------------`

- `ExifVersion`
- `FlashpixVersion`
- `ColorSpace`
- `ComponentsConfiguration`
- `CompressedBitsPerPixel`
- `PixelXDimension`
- `PixelYDimension`
- `MakerNote`
- `UserComment`
- `RelatedSoundFile`
- `DateTimeOriginal`
- `DateTimeDigitized`
- `SubSecTime`
- `SubSecTimeOriginal`
- `SubSecTimeDigitized`
- `ExposureTime`
- `FNumber`
- `ExposureProgram`
- `SpectralSensitivity`
- `ISOSpeedRatings`
- `OECF`
- `ShutterSpeedValue`
- `ApertureValue`
- `BrightnessValue`
- `ExposureBiasValue`
- `MaxApertureValue`
- `SubjectDistance`
- `MeteringMode`
- `LightSource`
- `Flash`
- `FocalLength`
- `SubjectArea`
- `FlashEnergy`
- `SpatialFrequencyResponse`
- `FocalPlaneXResolution`
- `FocalPlaneYResolution`
- `FocalPlaneResolutionUnit`
- `SubjectLocation`
- `ExposureIndex`
- `SensingMethod`
- `FileSource`
- `SceneType`
- `CFAPattern`
- `CustomRendered`
- `ExposureMode`
- `WhiteBalance`
- `DigitalZoomRatio`
- `FocalLengthIn35mmFilm`
- `SceneCaptureType`

- `GainControl`
- `Contrast`
- `Saturation`
- `Sharpness`
- `DeviceSettingDescription`
- `SubjectDistanceRange`
- `ImageUniqueID`

`-----------------------------`

- `GPSVersionID`
- `GPSLatitudeRef`
- `GPSLatitude`
- `GPSLongitudeRef`
- `GPSLongitude`
- `GPSAltitudeRef`
- `GPSAltitude`
- `GPSTimeStamp`
- `GPSSatellites`
- `GPSStatus`
- `GPSMeasureMode`
- `GPSDOP`
- `GPSSpeedRef`
- `GPSSpeed`
- `GPSTrackRef`
- `GPSTrack`
- `GPSImgDirectionRef`
- `GPSImgDirection`
- `GPSMapDatum`
- `GPSDestLatitudeRef`
- `GPSDestLatitude`
- `GPSDestLongitudeRef`
- `GPSDestLongitude`
- `GPSDestBearingRef`
- `GPSDestBearing`
- `GPSDestDistanceRef`
- `GPSDestDistance`
- `GPSProcessingMethod`
- `GPSAreaInformation`
- `GPSDateStamp`
- `GPSDifferential`

▼ image-exif-data [altova:]

**`image-exif-data(Base64BinaryString`** *`as string`* **`) as element?`** **`XP3.1`** **`XQ3.1`**

Takes a Base64-encoded JPEG image as its argument and returns an element called **`Exif`** that contains the Exif metadata of the image. The Exif metadata is created as attribute-value pairs of the `Exif` element. The attribute names are the Exif data tags found in the Base64 encoding. The list of Exif-specification tags is given below. If a vendor-specific tag is present in the Exif data, this tag and its value will also be returned as an attribute-value pair. Additional to the standard Exif metadata tags (*see list below*), Altova-specific attribute-value pairs are also generated. These Altova Exif attributes are listed below.

⊟ *Examples*

- To access any one attribute, use the function like this:
  `image-exif-data(//MyImages/Image20141130.01)/@GPSLatitude`
  `image-exif-data(//MyImages/Image20141130.01)/@Geolocation`
- To access all the attributes, use the function like this:
  `image-exif-data(//MyImages/Image20141130.01)/@*`
- To access the names of all the attributes, use the following expression:
  `for $i in image-exif-data(//MyImages/Image20141130.01)/@* return name($i)`
  This is useful to find out the names of the attributes returned by the function.

### ⊟ *Altova Exif Attribute: Geolocation*

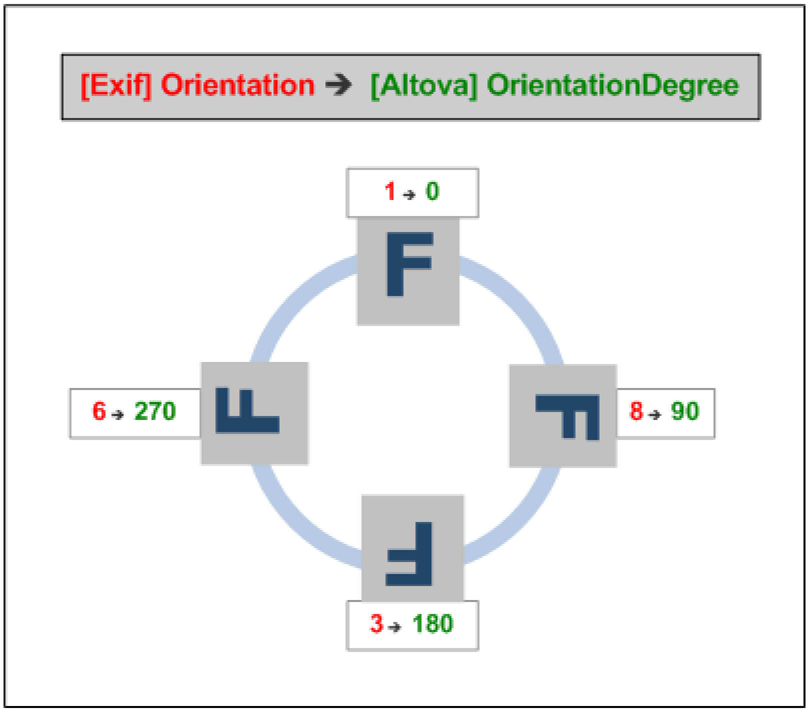
The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: **`GPSLatitude`**, **`GPSLatitudeRef`**, **`GPSLongitude`**, **`GPSLongitudeRef`**, with units added (*see table below*).

| GPSLatitude | GPSLatitudeRef | GPSLongitude | GPSLongitudeRef | Geolocation |
|---|---|---|---|---|
| 33 51 21.91 | S | 151 13 11.73 | E | 33°51'21.91"S 151°13'11.73"E |

### ⊟ *Altova Exif Attribute: OrientationDegree*

The Altova XPath/XQuery Engine generates the custom attribute **`OrientationDegree`** from the Exif metadata tag **`Orientation`**.

**`OrientationDegree`** translates the standard Exif tag **`Orientation`** from an integer value (**1**, **8**, **3**, or **6**) to the respective degree values of each (**0**, **90**, **180**, **270**), as shown in the figure below. Note that there are no translations of the **`Orientation`** values of **2**, **4**, **5**, **7**. (These orientations are obtained by flipping image 1 across its vertical center axis to get the image with a value of 2, and then rotating this image in 90-degree jumps clockwise to get the values of 7, 4, and 5, respectively).



⊟ *Listing of standard Exif meta tags*

- `ImageWidth`
- `ImageLength`
- `BitsPerSample`
- `Compression`
- `PhotometricInterpretation`
- `Orientation`
- `SamplesPerPixel`
- `PlanarConfiguration`
- `YCbCrSubSampling`
- `YCbCrPositioning`
- `XResolution`
- `YResolution`
- `ResolutionUnit`
- `StripOffsets`
- `RowsPerStrip`
- `StripByteCounts`
- `JPEGInterchangeFormat`
- `JPEGInterchangeFormatLength`
- `TransferFunction`
- `WhitePoint`
- `PrimaryChromaticities`
- `YCbCrCoefficients`
- `ReferenceBlackWhite`
- `DateTime`
- `ImageDescription`
- `Make`

- `Model`
- `Software`
- `Artist`
- `Copyright`

`------------------------------`

- `ExifVersion`
- `FlashpixVersion`
- `ColorSpace`
- `ComponentsConfiguration`
- `CompressedBitsPerPixel`
- `PixelXDimension`
- `PixelYDimension`
- `MakerNote`
- `UserComment`
- `RelatedSoundFile`
- `DateTimeOriginal`
- `DateTimeDigitized`
- `SubSecTime`
- `SubSecTimeOriginal`
- `SubSecTimeDigitized`
- `ExposureTime`
- `FNumber`
- `ExposureProgram`
- `SpectralSensitivity`
- `ISOSpeedRatings`
- `OECF`
- `ShutterSpeedValue`
- `ApertureValue`
- `BrightnessValue`
- `ExposureBiasValue`
- `MaxApertureValue`
- `SubjectDistance`
- `MeteringMode`
- `LightSource`
- `Flash`
- `FocalLength`
- `SubjectArea`
- `FlashEnergy`
- `SpatialFrequencyResponse`
- `FocalPlaneXResolution`
- `FocalPlaneYResolution`
- `FocalPlaneResolutionUnit`
- `SubjectLocation`
- `ExposureIndex`
- `SensingMethod`
- `FileSource`
- `SceneType`
- `CFAPattern`
- `CustomRendered`
- `ExposureMode`
- `WhiteBalance`
- `DigitalZoomRatio`
- `FocalLengthIn35mmFilm`
- `SceneCaptureType`

- `GainControl`
- `Contrast`
- `Saturation`
- `Sharpness`
- `DeviceSettingDescription`
- `SubjectDistanceRange`
- `ImageUniqueID`

`------------------------------`

- `GPSVersionID`
- `GPSLatitudeRef`
- `GPSLatitude`
- `GPSLongitudeRef`
- `GPSLongitude`
- `GPSAltitudeRef`
- `GPSAltitude`
- `GPSTimeStamp`
- `GPSSatellites`
- `GPSStatus`
- `GPSMeasureMode`
- `GPSDOP`
- `GPSSpeedRef`
- `GPSSpeed`
- `GPSTrackRef`
- `GPSTrack`
- `GPSImgDirectionRef`
- `GPSImgDirection`
- `GPSMapDatum`
- `GPSDestLatitudeRef`
- `GPSDestLatitude`
- `GPSDestLongitudeRef`
- `GPSDestLongitude`
- `GPSDestBearingRef`
- `GPSDestBearing`
- `GPSDestDistanceRef`
- `GPSDestDistance`
- `GPSProcessingMethod`
- `GPSAreaInformation`
- `GPSDateStamp`
- `GPSDifferential`

[ **Top** 1718 ]

## 30.1.1.4 XPath/XQuery Functions: Numeric

Altova's numeric extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | XP1 XP2 XP3.1 |
| *XSLT functions (used in XPath expressions in XSLT):* | XSLT1 XSLT2 XSLT3 |
| *XQuery functions (used in XQuery expressions in XQuery):* | XQ1 XQ3.1 |

## Auto-numbering functions

### ▼ generate-auto-number [altova:]

**generate-auto-number(ID** *as xs:string*, **StartsWith** *as xs:double*, **Increment** *as xs:double*, **ResetOnChange** *as xs:string*) as `xs:integer` XP1 XP2 XQ1 XP3.1 XQ3.1

Generates a number each time the function is called. The first number, which is generated the first time the function is called, is specified by the `StartsWith` argument. Each subsequent call to the function generates a new number, this number being incremented over the previously generated number by the value specified in the `Increment` argument. In effect, the `generate-auto-number` function creates a counter having a name specified by the `ID` argument, with this counter being incremented each time the function is called. If the value of the `ResetOnChange` argument changes from that of the previous function call, then the value of the number to be generated is reset to the `StartsWith` value. Auto-numbering can also be reset by using the `reset-auto-number` function.

⊟ *Examples*

- **`generate-auto-number`**`("ChapterNumber", 1, 1, "SomeString")` will return one number each time the function is called, starting with `1`, and incrementing by `1` with each call to the function. As long as the fourth argument remains `"SomeString"` in each subsequent call, the incrementing will continue. When the value of the fourth argument changes, the counter (called `ChapterNumber`) will reset to 1. The value of `ChapterNumber` can also be reset by a call to the `reset-auto-number` function, like this: `reset-auto-number("ChapterNumber")`.

### ▼ reset-auto-number [altova:]

**reset-auto-number(ID** *as xs:string*) XP1 XP2 XQ1 XP3.1 XQ3.1

This function resets the number of the auto-numbering counter named in the `ID` argument. The number is reset to the number specified by the `StartsWith` argument of the `generate-auto-number` function that created the counter named in the `ID` argument.

⊟ *Examples*

- **`reset-auto-number`**`("ChapterNumber")` resets the number of the auto-numbering counter named `ChapterNumber` that was created by the `generate-auto-number` function. The number is reset to the value of the `StartsWith` argument of the `generate-auto-number` function that created `ChapterNumber`.

[ Top1727 ]

## Numeric functions

▼ hex-string-to-integer [altova:]

**hex-string-to-integer(HexString** *as xs:string***) as xs:integer** XP3.1 XQ3.1

Takes a string argument that is the Base-16 equivalent of an integer in the decimal system (Base-10), and returns the decimal integer.

⊟ *Examples*

- `hex-string-to-integer('1')` returns `1`
- `hex-string-to-integer('9')` returns `9`
- `hex-string-to-integer('A')` returns `10`
- `hex-string-to-integer('B')` returns `11`
- `hex-string-to-integer('F')` returns `15`
- `hex-string-to-integer('G')` returns an error
- `hex-string-to-integer('10')` returns `16`
- `hex-string-to-integer('01')` returns `1`
- `hex-string-to-integer('20')` returns `32`
- `hex-string-to-integer('21')` returns `33`
- `hex-string-to-integer('5A')` returns `90`
- `hex-string-to-integer('USA')` returns an error

▼ integer-to-hex-string [altova:]

**integer-to-hex-string(Integer** *as xs:integer***) as xs:string** XP3.1 XQ3.1

Takes an integer argument and returns its Base-16 equivalent as a string.

⊟ *Examples*

- `integer-to-hex-string(1)` returns `'1'`
- `integer-to-hex-string(9)` returns `'9'`
- `integer-to-hex-string(10)` returns `'A'`
- `integer-to-hex-string(11)` returns `'B'`
- `integer-to-hex-string(15)` returns `'F'`
- `integer-to-hex-string(16)` returns `'10'`
- `integer-to-hex-string(32)` returns `'20'`
- `integer-to-hex-string(33)` returns `'21'`
- `integer-to-hex-string(90)` returns `'5A'`

[ Top1727 ]

## Number-formatting functions

▼ mt-format-number [altova:]

**mt-format-number(Number** *as xs:numeric*, **PictureString** *as xs:string***) as xs:string** XP3.1 XQ3.1

Takes a number as the first argument, formats it according to the second (`PictureString`) argument, and returns the formatted number as a string. This is useful for formatting difficult-to-read numbers into a format that is more reader-friendly. The picture string can also contain characters, such as currency symbols, and so can also be used to insert characters in the formatted output. If you wish to insert a zero at a digit position when no digit exists in the input number at that position, then use a zero in that digit position of the picture string (*see examples below*). If you do not wish to force a zero (or other character), use the hash symbol (#).

Digits before the decimal separator are never foreshortened. The decimal part of a number (to the right of the decimal separator) as well as the units digit (first digit to the left of the decimal separator) are rounded off if the picture string of the decimal part is shorter than the number of decimal places in the input number.

**Note:** The grouping separator and decimal separator in the formatted output on the mobile device will be those of the language being used on the mobile device.

⊟ *Examples*

- `mt-format-number(12.3, '$#0.00')` returns `$12.30`
- `mt-format-number(12.3, '$00.00')` returns `$12.30`
- `mt-format-number(12.3, '$0,000.00')` returns `$0,012.30`
- `mt-format-number(12.3, '$#,000.00')` returns `$012.30`
- `mt-format-number(1234.5, '$#,##0.00')` returns `$1,234.50`
- `mt-format-number(1234.5, '$#0.00')` returns `$1234.50`
- `mt-format-number(123.4, '$0')` returns `$123`
- `mt-format-number(1234.5, '$0')` returns `$1235`
- `mt-format-number(1234.54, '$0.0')` returns `$1234.5`
- `mt-format-number(1234.55, '$0.0')` returns `$1234.6`

[ Top 1727 ]

## 30.1.1.5 XPath/XQuery Functions: Schema

The Altova extension functions listed below return schema information. Given below are descriptions of the functions, together with (i) examples and (ii) a listing of schema components and their respective properties.

*Schema information from schema documents*

The function **`altova:schema`** has two arguments: one with zero arguments and the other with two arguments. The zero-argument function returns the whole schema. You can then, from this starting point, navigate into the schema to locate the schema components you want. The two-argument function returns a specific component kind that is identified by its QName. In both cases, the return value is a function. To navigate into the returned component, you must select a property of that specific component. If the property is a non-atomic item (that is, if it is a component), then you can navigate further by selecting a property of this component. If the selected property is an atomic item, then the value of the item is returned and you cannot navigate any further.

**Note:** In XPath expressions, the schema must be imported into the processing environment (for example, into XSLT) with the **`xslt:import-schema`** instruction. In XQuery expressions, the schema must be explicitly imported using a schema import.

*Schema information from XML nodes*

The function **`altova:type`** submits the node of an XML document and returns the node's type information from the PSVI.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | XP1 XP2 XP3.1 |
| *XSLT functions (used in XPath expressions in XSLT):* | XSLT1 XSLT2 XSLT3 |
| *XQuery functions (used in XQuery expressions in XQuery):* | XQ1 XQ3.1 |

### Schema (zero arguments)

**`altova:schema()`** `as (function(xs:string) as item()*)?` XP3.1 XQ3.1

Returns the `schema` component as a whole. You can navigate further into the **`schema`** component by selecting one of the **`schema`** component's properties.

- If this property is a component, you can navigate another step deeper by selecting one of this component's properties. This step can be repeated to navigate further into the schema.
- If the component is an atomic value, the atomic value is returned and you cannot navigate any deeper.

The properties of the **`schema`** component are:

```
"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"
```

The properties of all other component kinds (besides **`schema`**) are listed below.

**Note:** In XQuery expressions, the schema must be explicitly imported. In XPath expressions, the schema must have been imported into the processing environment, for example, into XSLT with the **`xslt:import`** instruction.

*Examples*

- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema()("type definitions")` `return $typedef ("name")` returns the names of all simple types or complex types in the schema

- `import schema "" at "C:\Test\ExpReport.xsd";`
  `altova:schema() ("type definitions")[1]("name")` returns the name of the first of all simple types or complex types in the schema

*Components and their properties*

⊟ Assertion

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Assertion" |
| test | XPath Property Record | |

⊟ Attribute Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Declaration" |
| name | string | Local name of the attribute |
| target namespace | string | Namespace URI of the attribute |
| type definition | Simple Type or Complex Type | |
| scope | A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group) | |
| value constraint | If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types | |
| inheritable | boolean | |

⊟ Attribute Group Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Group Definition" |
| name | string | Local name of the attribute group |
| target namespace | string | Namespace URI of the attribute group |
| attribute uses | Sequence of (Attribute Use) | |
| attribute wildcard | Optional Attribute Wildcard | |

### Attribute Use

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Use" |
| required | boolean | true if the attribute is required, false if optional |
| value constraint | See Attribute Declaration | |
| inheritable | boolean | |

### Attribute Wildcard

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

### Complex Type

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Complex Type" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| base type definition | Complex Type Definition | |
| final | Sequence of strings ("restriction"\|"extension") | |
| context | Empty sequence (not implemented) | |
| derivation method | string ("restriction"\|"extension") | |
| abstract | boolean | |
| attribute uses | Sequence of Attribute Use | |
| attribute wildcard | Optional Attribute Wildcard | |
| content type | function with properties: ("class":"Content Type", "variety":string ("element-only"\|"empty"\|"mixed"\|"simple"), particle: | |

| | | |
|---|---|---|
| | optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"\|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type) | |
| prohibited substitutions | Sequence of strings ("restriction"\|"extension") | |
| assertions | Sequence of Assertion | |

⊟ Element Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Complex Type" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| type definition | Simple Type or Complex Type | |
| type table | function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type) | |
| scope | function with properties ("class":"Scope", "variety": ("global"\|"local"), "parent": optional Complex Type) | |
| value constraint | see Attribute Declaration | |
| nillable | boolean | |
| identity-constraint definitions | Sequence of Identity Constraint | |
| substitution group affiliations | Sequence of Element Declaration | |
| substitution group exclusions | Sequence of strings ("restriction"\|"extension") | |
| disallowed substitutions | Sequence of strings ("restriction"\|"extension"\|"substitution") | |
| abstract | boolean | |

⊟ Element Wildcard

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": | |

| | | |
|---|---|---|
| | "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

Facet

| Property name | Property type | Property value |
|---|---|---|
| kind | string | The name of the facet, for example "minLength" or "enumeration" |
| value | depends on facet | The value of the facet |
| fixed | boolean | |
| typed-value | For the enumeration facet only, array(xs:anyAtomicType*) | An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type) |

Identity Constraint

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Identity-Constraint Definition" |
| name | string | Local name of the constraint |
| target namespace | string | Namespace URI of the constraint |
| identity-constraint category | string ("key"\|"unique"\|"keyRef") | |
| selector | XPath Property Record | |
| fields | Sequence of XPath Property Record | |
| referenced key | (For keyRef only): Identity Constraint | The corresponding key constraint |

Model Group

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Model Group" |
| compositor | string ("sequence"\|"choice"\|"all") | |
| particles | Sequence of Particle | |

### Model Group Definition

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Model Group Definition" |
| name | string | Local name of the model group |
| target namespace | string | Namespace URI of the model group |
| model group | Model Group | |

### Notation

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Notation Declaration" |
| name | string | Local name of the notation |
| target namespace | string | Namespace URI of the notation |
| system identifier | anyURI | |
| public identifier | string | |

### Particle

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Particle" |
| min occurs | integer | |
| max occurs | integer, or string("unbounded") | |
| term | Element Declaration, Element Wildcard, or ModelGroup | |

### Simple Type

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Simple Type Definition" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| final | Sequence of string("restriction"\|"extension"\|"list"\|"union") | |
| context | containing component | |
| base type definition | Simple Type | |
| facets | Sequence of Facet | |
| fundamental facets | Empty sequence (not implemented) | |

| | | |
|---|---|---|
| variety | string ("atomic"\|"list"\|"union") | |
| primitive type definition | Simple Type | |
| item type definition | (for list types only) Simple Type | |
| member type definitions | (for union types only) Sequence of Simple Type | |

Type Alternative

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Type Alternative" |
| test | XPath Property Record | |
| type definition | Simple Type or Complex Type | |

XPath Property Record

| Property name | Property type | Property value |
|---|---|---|
| namespace bindings | Sequence of functions with properties ("prefix": string, "namespace": anyURI) | |
| default namespace | anyURI | |
| base URI | anyURI | The static base URI of the XPath expression |
| expression | string | The XPath expression as a string |

Schema (two arguments)

**altova:schema(ComponentKind** *as xs:string*, **Name** *as xs:QName***) as (function(xs:string) as item()*)? XP3.1 XQ3.1**

Returns the component kind that is specified in the first argument which has a name that is the same as the name supplied in the second argument. You can navigate further by selecting one of the component's properties.

- If this property is a component, you can navigate another step deeper by selecting one of this component's properties. This step can be repeated to navigate further into the schema.
- If the component is an atomic value, the atomic value is returned and you cannot navigate any deeper.

**Note:** In XQuery expressions, the schema must be explicitly imported. In XPath expressions, the schema must have been imported into the processing environment, for example, into XSLT with the **`xslt:import`** instruction.

*Examples*

-
  ```
  import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
  ```

```
("content type")("particles")[3]!.("term")("kind")
```

returns the `kind` property of the term of the third `particles` component. This particles component is a descendant of the element declaration having a `QName` of `OrgChart`

-
  ```
  import schema "" at "C:\Test\ExpReport.xsd";
  let $typedef := altova:schema("type definition", xs:QName("emailType"))
  for $facet in $typedef ("facets")
  return [$facet ("kind"), $facet("value")]
  ```
  returns, for each **`facet`** of each **`emailType`** component, an array containing that facet's kind and value

*Components and their properties*

⊟ Assertion

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Assertion" |
| test | XPath Property Record | |

⊟ Attribute Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Declaration" |
| name | string | Local name of the attribute |
| target namespace | string | Namespace URI of the attribute |
| type definition | Simple Type or Complex Type | |
| scope | A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group) | |
| value constraint | If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types | |
| inheritable | boolean | |

⊟ Attribute Group Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Group Definition" |
| name | string | Local name of the attribute group |
| target namespace | string | Namespace URI of the attribute |

| | | |
|---|---|---|
| | | group |
| attribute uses | Sequence of (Attribute Use) | |
| attribute wildcard | Optional Attribute Wildcard | |

⊟ Attribute Use

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Use" |
| required | boolean | true if the attribute is required, false if optional |
| value constraint | See Attribute Declaration | |
| inheritable | boolean | |

⊟ Attribute Wildcard

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

⊟ Complex Type

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Complex Type" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| base type definition | Complex Type Definition | |
| final | Sequence of strings ("restriction"\|"extension") | |
| context | Empty sequence (not implemented) | |
| derivation method | string ("restriction"\|"extension") | |
| abstract | boolean | |
| attribute uses | Sequence of Attribute Use | |
| attribute wildcard | Optional Attribute Wildcard | |

| | | |
|---|---|---|
| content type | function with properties: ("class":"Content Type", "variety":string ("element-only"\|"empty"\|"mixed"\|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"\|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type) | |
| prohibited substitutions | Sequence of strings ("restriction"\|"extension") | |
| assertions | Sequence of Assertion | |

⊟ Element Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Complex Type" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| type definition | Simple Type or Complex Type | |
| type table | function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type) | |
| scope | function with properties ("class":"Scope", "variety": ("global"\|"local"), "parent": optional Complex Type) | |
| value constraint | see Attribute Declaration | |
| nillable | boolean | |
| identity-constraint definitions | Sequence of Identity Constraint | |
| substitution group affiliations | Sequence of Element Declaration | |
| substitution group exclusions | Sequence of strings ("restriction"\|"extension") | |
| disallowed substitutions | Sequence of strings ("restriction"\|"extension"\|"substitution") | |
| abstract | boolean | |

⊟ Element Wildcard

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

⊟ Facet

| Property name | Property type | Property value |
|---|---|---|
| kind | string | The name of the facet, for example "minLength" or "enumeration" |
| value | depends on facet | The value of the facet |
| fixed | boolean | |
| typed-value | For the enumeration facet only, array(xs:anyAtomicType*) | An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type) |

⊟ Identity Constraint

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Identity-Constraint Definition" |
| name | string | Local name of the constraint |
| target namespace | string | Namespace URI of the constraint |
| identity-constraint category | string ("key"\|"unique"\|"keyRef") | |
| selector | XPath Property Record | |
| fields | Sequence of XPath Property Record | |
| referenced key | (For keyRef only): Identity Constraint | The corresponding key constraint |

⊟ Model Group

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Model Group" |

| | | |
|---|---|---|
| compositor | string ("sequence"\|"choice"\|"all") | |
| particles | Sequence of Particle | |

### Model Group Definition

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Model Group Definition" |
| name | string | Local name of the model group |
| target namespace | string | Namespace URI of the model group |
| model group | Model Group | |

### Notation

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Notation Declaration" |
| name | string | Local name of the notation |
| target namespace | string | Namespace URI of the notation |
| system identifier | anyURI | |
| public identifier | string | |

### Particle

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Particle" |
| min occurs | integer | |
| max occurs | integer, or string("unbounded") | |
| term | Element Declaration, Element Wildcard, or ModelGroup | |

### Simple Type

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Simple Type Definition" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| final | Sequence of string("restriction"\|"extension"\|"list"\|"union") | |

| | | |
|---|---|---|
| context | containing component | |
| base type definition | Simple Type | |
| facets | Sequence of Facet | |
| fundamental facets | Empty sequence (not implemented) | |
| variety | string ("atomic"\|"list"\|"union") | |
| primitive type definition | Simple Type | |
| item type definition | (for list types only) Simple Type | |
| member type definitions | (for union types only) Sequence of Simple Type | |

Type Alternative

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Type Alternative" |
| test | XPath Property Record | |
| type definition | Simple Type or Complex Type | |

XPath Property Record

| Property name | Property type | Property value |
|---|---|---|
| namespace bindings | Sequence of functions with properties ("prefix": string, "namespace": anyURI) | |
| default namespace | anyURI | |
| base URI | anyURI | The static base URI of the XPath expression |
| expression | string | The XPath expression as a string |

### Type

```
altova:type(Node as item?) as (function(xs:string) as item()*)? XP3.1 XQ3.1
```

The function **`altova:type`** submits an element or attribute node of an XML document and returns the node's type information from the PSVI.

**Note:** The XML document must have a schema declaration so that the schema can be referenced.

*Examples*

-
  ```
  for $element in //Email
  let $type := altova:type($element)
  return $type
  ```
  returns a function that contains the `Email` node's type information

-
  ```
  for $element in //Email
  ```

```
let $type := altova:type($element)
return $type ("kind")
```

takes the `Email` node's type component (Simple Type or Complex Type) and returns the value of the component's **`kind`** property

The **`"_props"`** parameter returns the properties of the selected component. For example:

-
  ```
  for $element in //Email
  let $type := altova:type($element)
  return ($type ("kind"), $type ("_props"))
  ```
  takes the `Email` node's type component (Simple Type or Complex Type) and returns (i) the value of the component's **`kind`** property, and then (ii) the properties of that component.

*Components and their properties*

Assertion

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Assertion" |
| test | XPath Property Record | |

Attribute Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Declaration" |
| name | string | Local name of the attribute |
| target namespace | string | Namespace URI of the attribute |
| type definition | Simple Type or Complex Type | |
| scope | A function with properties ("class":"Scope", "variety": "global" or "local", "parent": the containing Complex Type or Attribute Group) | |
| value constraint | If present, a function with properties ("class": "Value Constraint", "variety": "fixed" or "default", "value": atomic value, "lexical form": string. Note that the "value" property is not available for namespace-sensitive types | |
| inheritable | boolean | |

Attribute Group Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Group Definition" |

| | | |
|---|---|---|
| name | string | Local name of the attribute group |
| target namespace | string | Namespace URI of the attribute group |
| attribute uses | Sequence of (Attribute Use) | |
| attribute wildcard | Optional Attribute Wildcard | |

### Attribute Use

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Attribute Use" |
| required | boolean | true if the attribute is required, false if optional |
| value constraint | See Attribute Declaration | |
| inheritable | boolean | |

### Attribute Wildcard

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

### Complex Type

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Complex Type" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| base type definition | Complex Type Definition | |
| final | Sequence of strings ("restriction"\|"extension") | |
| context | Empty sequence (not implemented) | |
| derivation method | string ("restriction"\|"extension") | |
| abstract | boolean | |

| | | |
|---|---|---|
| attribute uses | Sequence of Attribute Use | |
| attribute wildcard | Optional Attribute Wildcard | |
| content type | function with properties: ("class":"Content Type", "variety":string ("element-only"\|"empty"\|"mixed"\|"simple"), particle: optional Particle, "open content": function with properties ("class":"Open Content", "mode": string ("interleave"\|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type) | |
| prohibited substitutions | Sequence of strings ("restriction"\|"extension") | |
| assertions | Sequence of Assertion | |

⊟ Element Declaration

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Complex Type" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| type definition | Simple Type or Complex Type | |
| type table | function with properties ("class":"Type Table", "alternatives": sequence of Type Alternative, "default type definition": Simple Type or Complex Type) | |
| scope | function with properties ("class":"Scope", "variety": ("global"\|"local"), "parent": optional Complex Type) | |
| value constraint | see Attribute Declaration | |
| nillable | boolean | |
| identity-constraint definitions | Sequence of Identity Constraint | |
| substitution group affiliations | Sequence of Element Declaration | |
| substitution group exclusions | Sequence of strings ("restriction"\|"extension") | |
| disallowed substitutions | Sequence of strings ("restriction"\|"extension"\|"substitution") | |
| abstract | boolean | |

Element Wildcard

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Wildcard" |
| namespace constraint | function with properties ("class": "Namespace Constraint", "variety": "any"\|"enumeration"\|"not", "namespaces": sequence of xs:anyURI, "disallowed names": list containing QNames and/or the strings "defined" and "definedSiblings" | |
| process contents | string ("strict"\|"lax"\|"skip") | |

Facet

| Property name | Property type | Property value |
|---|---|---|
| kind | string | The name of the facet, for example "minLength" or "enumeration" |
| value | depends on facet | The value of the facet |
| fixed | boolean | |
| typed-value | For the enumeration facet only, array(xs:anyAtomicType*) | An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the "value" property is a sequence of strings, regardless of the actual type) |

Identity Constraint

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Identity-Constraint Definition" |
| name | string | Local name of the constraint |
| target namespace | string | Namespace URI of the constraint |
| identity-constraint category | string ("key"\|"unique"\|"keyRef") | |
| selector | XPath Property Record | |
| fields | Sequence of XPath Property Record | |
| referenced key | (For keyRef only): Identity Constraint | The corresponding key constraint |

Model Group

| Property name | Property type | Property value |
|---|---|---|

| | | |
|---|---|---|
| kind | string | "Model Group" |
| compositor | string ("sequence"\|"choice"\|"all") | |
| particles | Sequence of Particle | |

Model Group Definition

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Model Group Definition" |
| name | string | Local name of the model group |
| target namespace | string | Namespace URI of the model group |
| model group | Model Group | |

Notation

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Notation Declaration" |
| name | string | Local name of the notation |
| target namespace | string | Namespace URI of the notation |
| system identifier | anyURI | |
| public identifier | string | |

Particle

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Particle" |
| min occurs | integer | |
| max occurs | integer, or string("unbounded") | |
| term | Element Declaration, Element Wildcard, or ModelGroup | |

Simple Type

| Property name | Property type | Property value |
|---|---|---|
| kind | string | "Simple Type Definition" |
| name | string | Local name of the type (empty if anonymous) |
| target namespace | string | Namespace URI of the type (empty if anonymous) |
| final | Sequence of string("restriction"\|"extension"\|"list"\|"unio | |

| | | |
|---|---|---|
| | n") | |
| context | containing component | |
| base type definition | Simple Type | |
| facets | Sequence of Facet | |
| fundamental facets | Empty sequence (not implemented) | |
| variety | string ("atomic"\|"list"\|"union") | |
| primitive type definition | Simple Type | |
| item type definition | (for list types only) Simple Type | |
| member type definitions | (for union types only) Sequence of Simple Type | |

Type Alternative

| **Property name** | **Property type** | **Property value** |
|---|---|---|
| kind | string | "Type Alternative" |
| test | XPath Property Record | |
| type definition | Simple Type or Complex Type | |

XPath Property Record

| **Property name** | **Property type** | **Property value** |
|---|---|---|
| namespace bindings | Sequence of functions with properties ("prefix": string, "namespace": anyURI) | |
| default namespace | anyURI | |
| base URI | anyURI | The static base URI of the XPath expression |
| expression | string | The XPath expression as a string |

## 30.1.1.6 XPath/XQuery Functions: Sequence

Altova's sequence extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-`**

**`years-to-date [altova:]`**.

- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | **XP1** **XP2** **XP3.1** |
| *XSLT functions (used in XPath expressions in XSLT):* | **XSLT1** **XSLT2** **XSLT3** |
| *XQuery functions (used in XQuery expressions in XQuery):* | **XQ1** **XQ3.1** |

▼ attributes [altova:]

**`attributes(AttributeName`** *`as xs:string`***`)`** **`as attribute()*`** **XP3.1** **XQ3.1**

Returns all attributes that have a local name which is the same as the name supplied in the input argument, `AttributeName`. The search is case-sensitive and conducted along the `attribute::` axis. This means that the context node must be the parent element node.

⊟ *Examples*

- **`attributes`**`("MyAttribute")` returns `MyAttribute()*`

**`attributes(AttributeName`** *`as xs:string`*`,` **`SearchOptions`** *`as xs:string`***`)`** **`as attribute()*`** **XP3.1** **XQ3.1**

Returns all attributes that have a local name which is the same as the name supplied in the input argument, `AttributeName`. The search is case-sensitive and conducted along the `attribute::` axis. The context node must be the parent element node. The second argument is a string containing option flags. Available flags are:

`r` = switches to a regular-expression search; `AttributeName` must then be a regular-expression search string;

`f` = If this option is specified, then `AttributeName` provides a full match; otherwise `AttributeName` need only partially match an attribute name to return that attribute. For example: if `f` is not specified, then `MyAtt` will return `MyAttribute`;

`i` = switches to a case-insensitive search;

`p` = includes the namespace prefix in the search; `AttributeName` should then contain the namespace prefix, for example: `altova:MyAttribute`.

The flags can be written in any order. Invalid flags will generate errors. One or more flags can be omitted. The empty string is allowed, and will produce the same effect as the function having only one argument (*previous signature*). However, an empty sequence is not allowed as the second argument.

⊟ *Examples*

- **`attributes`**`("MyAttribute", "rfip")` returns `MyAttribute()*`
- **`attributes`**`("MyAttribute", "pri")` returns `MyAttribute()*`
- **`attributes`**`("MyAtt", "rip")` returns `MyAttribute()*`
- **`attributes`**`("MyAttributes", "rfip")` returns no match
- **`attributes`**`("MyAttribute", "")` returns `MyAttribute()*`
- **`attributes`**`("MyAttribute", "Rip")` returns an unrecognized-flag error.
- **`attributes`**`("MyAttribute", )` returns a missing-second-argument error.

▼ elements [altova:]

**`elements(ElementName`** *`as xs:string`***`)`** **`as element()*`** **XP3.1** **XQ3.1**

Returns all elements that have a local name which is the same as the name supplied in the input argument, `ElementName`. The search is case-sensitive and conducted along the `child::` axis. The context node must be the parent node of the element/s being searched for.

⊟ *Examples*

- `elements("MyElement")` returns `MyElement()*`

**`elements(ElementName`** *`as xs:string`*, **`SearchOptions`** *`as xs:string`*) **`as element()*`** **XP3.1** **XQ3.1**

Returns all elements that have a local name which is the same as the name supplied in the input argument, `ElementName`. The search is case-sensitive and conducted along the `child::` axis. The context node must be the parent node of the element/s being searched for. The second argument is a string containing option flags. Available flags are:

`r` **=** switches to a regular-expression search; `ElementName` must then be a regular-expression search string;

`f` **=** If this option is specified, then `ElementName` provides a full match; otherwise `ElementName` need only partially match an element name to return that element. For example: if `f` is not specified, then `MyElem` will return `MyElement`;

`i` **=** switches to a case-insensitive search;

`p` **=** includes the namespace prefix in the search; `ElementName` should then contain the namespace prefix, for example: `altova:MyElement`.

The flags can be written in any order. Invalid flags will generate errors. One or more flags can be omitted. The empty string is allowed, and will produce the same effect as the function having only one argument (*previous signature*). However, an empty sequence is not allowed.

⊟ *Examples*

- `elements("MyElement", "rip")` returns `MyElement()*`
- `elements("MyElement", "pri")` returns `MyElement()*`
- `elements("MyElement", "")` returns `MyElement()*`
- `elements("MyElem", "rip")` returns `MyElement()*`
- `elements("MyElements", "rfip")` returns no match
- `elements("MyElement", "Rip")` returns an unrecognized-flag error.
- `elements("MyElement", )` returns a missing-second-argument error.

▼ find-first [altova:]

**`find-first((Sequence`** *`as item()*`*), **`(Condition( Sequence-Item`** *`as xs:boolean`*)) **`as item()?`** **XP3.1** **XQ3.1**

This function takes two arguments. The first argument is a sequence of one or more items of any datatype. The second argument, `Condition`, is a reference to an XPath function that takes one argument (has an arity of **`1`**) and returns a boolean. Each item of **`Sequence`** is submitted, in turn, to the function referenced in `Condition`. (*Remember:* This function takes a single argument.) The first **`Sequence`** item that causes the function in **`Condition`** to evaluate to **`true()`** is returned as the result of **`find-first`**, and the iteration stops.

⊟ *Examples*

- `find-first(5 to 10, function($a) {$a mod 2 = 0})` returns `xs:integer 6`

  The **`Condition`** argument references the XPath 3.0 inline function, **`function()`**, which declares an inline function named **`$a`** and then defines it. Each item in the **`Sequence`** argument of **`find-first`** is passed, in turn, to **`$a`** as its input value. The input value is tested on the condition in the function definition (`$a mod 2 = 0`). The first input value to satisfy this condition is returned as the result of

**find-first** (in this case **6**).

- `find-first((1 to 10), (function($a) {$a+3=7}))` returns `xs:integer 4`

*Further examples*
If the file **C:\Temp\Customers.xml** exists:

- `find-first( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` returns `xs:string C:\Temp\Customers.xml`

If the file **C:\Temp\Customers.xml** does not exist, and **http://www.altova.com/index.html** exists:

- `find-first( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` returns `xs:string http://www.altova.com/index.html`

If the file **C:\Temp\Customers.xml** does not exist, and **http://www.altova.com/index.html** also does not exist:

- `find-first( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` returns no result

*Notes about the examples given above*

- The XPath 3.0 function, `doc-available`, takes a single string argument, which is used as a URI, and returns `true` if a document node is found at the submitted URI. (The document at the submitted URI must therefore be an XML document.)
- The `doc-available` function can be used for **Condition**, the second argument of `find-first`, because it takes only one argument (arity=1), because it takes an `item()` as input (a string which is used as a URI), and returns a boolean value.
- Notice that the `doc-available` function is only referenced, not called. The `#1` suffix that is attached to it indicates a function with an arity of 1. In its entirety `doc-available#1` simply means: *Use the doc-availabe() function that has arity=1, passing to it as its single argument, in turn, each of the items in the first sequence*. As a result, each of the two strings will be passed to **doc-available()**, which uses the string as a URI and tests whether a document node exists at the URI. If one does, the **doc-available()** evaluates to `true()` and that string is returned as the result of the **find-first** function. *Note about the doc-available() function: Relative paths are resolved relative to the the current base URI, which is by default the URI of the XML document from which the function is loaded.*

▼ find-first-combination [altova:]

**find-first-combination((Seq-01** *as item()**), **(Seq-02** *as item()**), **(Condition( Seq-01-Item, Seq-02-Item** *as xs:boolean*)) **as item()*** XP3.1 XQ3.1
This function takes three arguments:

- The first two arguments, **Seq-01** and **Seq-02**, are sequences of one or more items of any datatype.
- The third argument, **Condition**, is a reference to an XPath function that takes two arguments (has

an arity of **2**) and returns a boolean.

The items of **Seq-01** and **Seq-02** are passed in ordered pairs (one item from each sequence making up a pair) as the arguments of the function in **Condition**. The pairs are ordered as follows.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X1 Y2), (X1 Y3) ... (X1 Yn), (X2 Y1), (X2 Y2) ... (Xn Yn)
```

The first ordered pair that causes the **Condition** function to evaluate to **true()** is returned as the result of **find-first-combination**. Note that: (i) If the **Condition** function iterates through the submitted argument pairs and does not once evaluate to **true()**, then **find-first-combination** returns *No results*; (ii) The result of **find-first-combination** will always be a pair of items (of any datatype) or no item at all.

⊟ *Examples*

- **find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) returns the sequence of xs:integers (11, 21)
- **find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) returns the sequence of xs:integers (11, 22)
- **find-first-combination**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 34}) returns the sequence of xs:integers (11, 23)

▼ find-first-pair [altova:]

**find-first-pair((Seq-01** *as item()**), (**Seq-02** *as item()**), (**Condition( Seq-01-Item, Seq-02-Item** *as xs:boolean*)) **as item()*** XP3.1 XQ3.1

This function takes three arguments:

- The first two arguments, **Seq-01** and **Seq-02**, are sequences of one or more items of any datatype.
- The third argument, **Condition**, is a reference to an XPath function that takes two arguments (has an arity of **2**) and returns a boolean.

The items of **Seq-01** and **Seq-02** are passed in ordered pairs as the arguments of the function in **Condition**. The pairs are ordered as follows.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

The first ordered pair that causes the **Condition** function to evaluate to **true()** is returned as the result of **find-first-pair**. Note that: (i) If the **Condition** function iterates through the submitted argument pairs and does not once evaluate to **true()**, then **find-first-pair** returns *No results*; (ii) The result of **find-first-pair** will always be a pair of items (of any datatype) or no item at all.

⊟ *Examples*

- **find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32}) returns the sequence of xs:integers (11, 21)
- **find-first-pair**(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33}) returns *No results*

Notice from the two examples above that the ordering of the pairs is: `(11, 21) (12, 22) (13, 23)...(20, 30)`. This is why the second example returns *No results* (because no ordered pair gives a sum of 33).

### find-first-pair-pos [altova:]

**`find-first-pair-pos((Seq-01`** *`as item()*`*`**), (Seq-02**` *`as item()*`*`**), (Condition( Seq-01-Item, Seq-02-Item**` *`as xs:boolean`*`**)) as xs:integer**` **XP3.1** **XQ3.1**

This function takes three arguments:

- The first two arguments, **`Seq-01`** and **`Seq-02`**, are sequences of one or more items of any datatype.
- The third argument, **`Condition`**, is a reference to an XPath function that takes two arguments (has an arity of **`2`**) and returns a boolean.

The items of **`Seq-01`** and **`Seq-02`** are passed in ordered pairs as the arguments of the function in **`Condition`**. The pairs are ordered as follows.

```
If   Seq-01 = X1, X2, X3 ... Xn
And  Seq-02 = Y1, Y2, Y3 ... Yn
Then (X1 Y1), (X2 Y2), (X3 Y3) ... (Xn Yn)
```

The index position of the first ordered pair that causes the **`Condition`** function to evaluate to **`true()`** is returned as the result of **`find-first-pair-pos`**. Note that if the **`Condition`** function iterates through the submitted argument pairs and does not once evaluate to **`true()`**, then **`find-first-pair-pos`** returns *No results*.

*Examples*

- `find-first-pair-pos(11 to 20, 21 to 30, function($a, $b) {$a+$b = 32})` returns `1`
- `find-first-pair-pos(11 to 20, 21 to 30, function($a, $b) {$a+$b = 33})` returns *No results*

Notice from the two examples above that the ordering of the pairs is: `(11, 21) (12, 22) (13, 23)...(20, 30)`. In the first example, the first pair causes the **`Condition`** function to evaluate to **`true()`**, and so its index position in the sequence, **`1`**, is returned. The second example returns *No results* because no pair gives a sum of 33.

### find-first-pos [altova:]

**`find-first-pos((Sequence`** *`as item()*`*`**), (Condition( Sequence-Item**` *`as xs:boolean`*`**)) as xs:integer**` **XP3.1** **XQ3.1**

This function takes two arguments. The first argument is a sequence of one or more items of any datatype. The second argument, `Condition`, is a reference to an XPath function that takes one argument (has an arity of **`1`**) and returns a boolean. Each item of **`Sequence`** is submitted, in turn, to the function referenced in `Condition`. (*Remember:* This function takes a single argument.) The first **`Sequence`** item that causes the function in **`Condition`** to evaluate to **`true()`** has its index position in **`Sequence`** returned as the result of **`find-first-pos`**, and the iteration stops.

⊟ *Examples*

- `find-first-pos(5 to 10, function($a) {$a mod 2 = 0})` returns `xs:integer 2`

The **`Condition`** argument references the XPath 3.0 inline function, **`function()`**, which declares an inline function named **`$a`** and then defines it. Each item in the **`Sequence`** argument of **`find-first-pos`** is passed, in turn, to **`$a`** as its input value. The input value is tested on the condition in the function definition (`$a mod 2 = 0`). The index position in the sequence of the first input value to satisfy this condition is returned as the result of **`find-first-pos`** (in this case **`2`**, since **`6`**, the first value (in the sequence) to satisfy the condition, is at index position **`2`** in the sequence).

- `find-first-pos((2 to 10), (function($a) {$a+3=7}))` returns `xs:integer 3`

*Further examples*

If the file **`C:\Temp\Customers.xml`** exists:

- `find-first-pos( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` returns `1`

If the file **`C:\Temp\Customers.xml`** does not exist, and **`http://www.altova.com/index.html`** exists:

- `find-first-pos( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` returns `2`

If the file **`C:\Temp\Customers.xml`** does not exist, and **`http://www.altova.com/index.html`** also does not exist:

- `find-first-pos( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )` returns no result

*Notes about the examples given above*

- The XPath 3.0 function, `doc-available`, takes a single string argument, which is used as a URI, and returns `true` if a document node is found at the submitted URI. (The document at the submitted URI must therefore be an XML document.)
- The `doc-available` function can be used for **`Condition`**, the second argument of `find-first-pos`, because it takes only one argument (arity=1), because it takes an `item()` as input (a string which is used as a URI), and returns a boolean value.
- Notice that the `doc-available` function is only referenced, not called. The `#1` suffix that is attached to it indicates a function with an arity of 1. In its entirety `doc-available#1` simply means: *Use the doc-availabe() function that has arity=1, passing to it as its single argument, in turn, each of the items in the first sequence*. As a result, each of the two strings will be passed to **`doc-available()`**, which uses the string as a URI and tests whether a document node exists at the URI. If one does, the **`doc-available()`** function evaluates to `true()` and the index position of that string in the sequence is returned as the result of the **`find-first-pos`** function. *Note about the doc-available() function: Relative paths are resolved relative to the the current base URI, which is by default the URI of the XML document from which the function is loaded.*

▼ for-each-attribute-pair [altova:]

**for-each-attribute-pair(Seq1** *as element()?*, **Seq2** *as element()?*, **Function** *as function()*) **as item()*** XP3.1 XQ3.1

The first two arguments identify two elements, the attributes of which are used to build attribute pairs, where one attribute of a pair is obtained from the first element and the other attribute is obtained from the second element. Attribute pairs are selected on the basis of having the same name, and the pairs are ordered alphabetically (on their names) into a set. If, for one attribute no corresponding attribute on the other element exists, then the pair is "disjoint", meaning that it consists of one member only. The function item (third argument `Function`) is applied separately to each pair in the sequence of pairs (joint and disjoint), resulting in an output that is a sequence of items.

⊟ *Examples*

- `for-each-attribute-pair(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b})` returns ...

  `(2, 4, 6)` if
  ```
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />
  ```

  `(2, 4, 6)` if
  ```
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

  `(2, 6)` if
  ```
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

  *Note: The result* `(2, 6)` *is obtained by way of the following action:* `(1+1, ()+2, 3+3, 4+())`. *If one of the operands is the empty sequence, as in the case of items 2 and 4, then the result of the addition is an empty sequence.*

- `for-each-attribute-pair(/Example/Test-A, /Example/Test-B, concat#2)` returns ...

  `(11, 22, 33)` if
  ```
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />
  ```

  `(11, 2, 33, 4)` if
  ```
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

▼ for-each-combination [altova:]

**for-each-combination(FirstSequence** *as item()**, **SecondSequence** *as item()**, **Function($i,$j){$i || $j} ) as item()*** XP3.1 XQ3.1

The items of the two sequences in the first two arguments are combined so that each item of the first sequence is combined, in order, once with each item of the second sequence. The function given as the third argument is applied to each combination in the resulting sequence, resulting in an output that is a sequence of items (*see example*).

⊟ *Examples*

- `for-each-combination( ('a', 'b', 'c'), ('1', '2', '3'), function($i, $j){$i || $j} )` returns `('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3')`

▼ for-each-matching-attribute-pair [altova:]

**`for-each-matching-attribute-pair(Seq1`** *`as element()?`*, **`Seq2`** *`as element()?`*, **`Function`** *`as function()`***`)`** `as item()*` **XP3.1** **XQ3.1**

The first two arguments identify two elements, the attributes of which are used to build attribute pairs, where one attribute of a pair is obtained from the first element and the other attribute is obtained from the second element. Attribute pairs are selected on the basis of having the same name, and the pairs are ordered alphabetically (on their names) into a set. If, for one attribute no corresponding attribute on the other element exists, then no pair is built. The function item (third argument `Function`) is applied separately to each pair in the sequence of pairs, resulting in an output that is a sequence of items.

⊟ *Examples*

- `for-each-matching-attribute-pair(/Example/Test-A, /Example/Test-B, function($a, $b){$a+b})` returns ...

  ```
  (2, 4, 6) if
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (2, 4, 6) if
  <Test-A att2="2" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />

  (2, 6) if
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att3="1" />
  ```

- `for-each-matching-attribute-pair(/Example/Test-A, /Example/Test-B, concat#2)` returns ...

  ```
  (11, 22, 33) if
  <Test-A att1="1" att2="2" att3="3" />
  <Test-B att1="1" att2="2" att3="3" />

  (11, 33) if
  <Test-A att4="4" att1="1" att3="3" />
  <Test-B att3="3" att2="2" att1="1" />
  ```

▼ substitute-empty [altova:]

**`substitute-empty(FirstSequence`** *`as item()*`*, **`SecondSequence`** *`as item()`***`)`** `as item()*` **XP3.1** **XQ3.1**

If `FirstSequence` is empty, returns `SecondSequence`. If `FirstSequence` is not empty, returns `FirstSequence`.

⊟ *Examples*

- `substitute-empty( (1,2,3), (4,5,6) )` returns `(1,2,3)`
- `substitute-empty( (), (4,5,6) )` returns `(4,5,6)`

## 30.1.1.7 XPath/XQuery Functions: String

Altova's string extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | **XP1** **XP2** **XP3.1** |
| *XSLT functions (used in XPath expressions in XSLT):* | **XSLT1** **XSLT2** **XSLT3** |
| *XQuery functions (used in XQuery expressions in XQuery):* | **XQ1** **XQ3.1** |

▼ camel-case [altova:]

**`camel-case(InputString`** *`as xs:string`***`)`** **`as xs:string`** **XP3.1** **XQ3.1**

Returns the input string **`InputString`** in CamelCase. The string is analyzed using the regular expression `'\s'` (which is a shortcut for the whitespace character). The first non-whitespace character after a whitespace or sequence of consecutive whitespaces is capitalized. The first character in the output string is capitalized.

⊟ *Examples*

- `camel-case("max")` returns `Max`
- `camel-case("max max")` returns `Max Max`
- `camel-case("file01.xml")` returns `File01.xml`
- `camel-case("file01.xml file02.xml")` returns `File01.xml File02.xml`
- `camel-case("file01.xml   file02.xml")` returns `File01.xml   File02.xml`
- `camel-case("file01.xml -file02.xml")` returns `File01.xml -file02.xml`

**`camel-case(InputString`** *`as xs:string`*, **`SplitChars`** *`as xs:string`*, **`IsRegex`** *`as xs:boolean`***`)`** **`as xs:string`** **XP3.1** **XQ3.1**

Converts the input string **`InputString`** to camel case by using **`SplitChars`** to determine the character/s that trigger the next capitalization. **`SplitChars`** is used as a regular expression when **`IsRegex = true()`**, or as plain characters when **`IsRegex = false()`**. The first character in the output string is capitalized.

⊟ *Examples*

- `camel-case("setname getname", "set|get", true())` returns `setName getName`
- `camel-case("altova\documents\testcases", "\", false())` returns `Altova\Documents\Testcases`

▼ char [altova:]

**char(Position** *as xs:integer***) as xs:string** XP3.1 XQ3.1

Returns a string containing the character at the position specified by the `Position` argument, in the string obtained by converting the value of the context item to `xs:string`. The result string will be empty if no character exists at the index submitted by the `Position` argument.

⊟ *Examples*

If the context item is **`1234ABCD`**:

- `char(2)` returns `2`
- `char(5)` returns `A`
- `char(9)` returns the empty string.
- `char(-2)` returns the empty string.

**char(InputString** *as xs:string***, Position** *as xs:integer***) as xs:string** XP3.1 XQ3.1

Returns a string containing the character at the position specified by the `Position` argument, in the string submitted as the `InputString` argument. The result string will be empty if no character exists at the index submitted by the `Position` argument.

⊟ *Examples*

- `char("2014-01-15", 5)` returns `-`
- `char("USA", 1)` returns `U`
- `char("USA", 10)` returns the empty string.
- `char("USA", -2)` returns the empty string.

▼ create-hash-from-string[altova:]

**create-hash-from-string(InputString** *as xs:string***) as xs:string** XP2 XQ1 XP3.1 XQ3.1

**create-hash-from-string(InputString** *as xs:string***, HashAlgo** *as xs:string***) as xs:string** XP2 XQ1 XP3.1 XQ3.1

Generates a hash string from `InputString` by using the hashing algorithm specified by the `HashAlgo` argument. The following hashing algorithms may be specified (in upper or lower case): **`MD5`**, **`SHA-1`**, **`SHA-224`**, **`SHA-256`**, **`SHA-384`**, **`SHA-512`**. If the second argument is not specified (*see the first signature above*), then the **`SHA-256`** hashing algorithm is used. Note that Windows clients do not support **`SHA-256`**.

⊟ *Examples*

- `create-hash-from-string('abc')` returns a hash string generated by using the **`SHA-256`** hashing algorithm.
- `create-hash-from-string('abc', 'md5')` returns a hash string generated by using the **`MD5`** hashing algorithm.
- `create-hash-from-string('abc', 'MD5')` returns a hash string generated by using the **`MD5`** hashing algorithm.

▼ first-chars [altova:]

**first-chars(X-Number** *as xs:integer***) as xs:string** XP3.1 XQ3.1

Returns a string containing the first `X-Number` of characters of the string obtained by converting the value of the context item to `xs:string`.

⊟ *Examples*

If the context item is **1234ABCD**:

- `first-chars(2)` returns `12`
- `first-chars(5)` returns `1234A`
- `first-chars(9)` returns `1234ABCD`

**`first-chars(InputString`** *`as xs:string`*, **`X-Number`** *`as xs:integer`*) **`as xs:string`** **XP3.1** **XQ3.1**
Returns a string containing the first `X-Number` of characters of the string submitted as the `InputString` argument.

⊟ *Examples*

- `first-chars("2014-01-15", 5)` returns `2014-`
- `first-chars("USA", 1)` returns `U`

▼ format-string [altova:]

**`format-string(InputString`** *`as xs:string`*, **`FormatSequence`** *`as item()*`*) **`as xs:string`** **XP3.1** **XQ3.1**
The input string (first argument) contains positional parameters (`%1`, `%2`, etc). Each parameter is replaced by the string item that is located at the corresponding position in the format sequence (submitted as the second argument). So the first item in the format sequence replaces the positional parameter `%1`, the second item replaces `%2`, and so on. The function returns this formatted string that contains the replacements. If no string exists for a positional parameter, then the positional parameter itself is returned. This happens when the index of a positional parameter is greater than the number of items in the format sequence.

⊟ *Examples*

- `format-string('Hello %1, %2, %3', ('Jane','John','Joe'))` returns `"Hello Jane, John, Joe"`
- `format-string('Hello %1, %2, %3', ('Jane','John','Joe', 'Tom'))` returns `"Hello Jane, John, Joe"`
- `format-string('Hello %1, %2, %4', ('Jane','John','Joe', 'Tom'))` returns `"Hello Jane, John, Tom"`
- `format-string('Hello %1, %2, %4', ('Jane','John','Joe'))` returns `"Hello Jane, John, %4"`

▼ last-chars [altova:]

**`last-chars(X-Number`** *`as xs:integer`*) **`as xs:string`** **XP3.1** **XQ3.1**
Returns a string containing the last `X-Number` of characters of the string obtained by converting the value of the context item to `xs:string`.

⊟ *Examples*

If the context item is **1234ABCD**:

- `last-chars(2)` returns `CD`
- `last-chars(5)` returns `4ABCD`
- `last-chars(9)` returns `1234ABCD`

**`last-chars(InputString`** *`as xs:string`*, **`X-Number`** *`as xs:integer`*) **`as xs:string`** **XP3.1** **XQ3.1**

Returns a string containing the last `X-Number` of characters of the string submitted as the `InputString` argument.

⊟ *Examples*

- `last-chars("2014-01-15", 5)` returns `01-15`
- `last-chars("USA", 10)` returns `USA`

▼ pad-string-left [altova:]

**`pad-string-left(StringToPad`** *`as xs:string`*`,` **`StringLength`** *`as xs:integer`*`,` **`PadCharacter`** *`as xs:string`***`) as xs:string`** **XP3.1** **XQ3.1**

The `PadCharacter` argument is a single character. It is padded to the left of the string to increase the number of characters in `StringToPad` so that this number equals the integer value of the `StringLength` argument. The `StringLength` argument can have any integer value (positive or negative), but padding will occur only if the value of `StringLength` is greater than the number of characters in `StringToPad`. If `StringToPad`. has more characters than the value of `StringLength`, then `StringToPad` is left unchanged.

⊟ *Examples*

- `pad-string-left('AP', 1, 'Z')` returns `'AP'`
- `pad-string-left('AP', 2, 'Z')` returns `'AP'`
- `pad-string-left('AP', 3, 'Z')` returns `'ZAP'`
- `pad-string-left('AP', 4, 'Z')` returns `'ZZAP'`
- `pad-string-left('AP', -3, 'Z')` returns `'AP'`
- `pad-string-left('AP', 3, 'YZ')` returns a pad-character-too-long error

▼ pad-string-right [altova:]

**`pad-string-right(StringToPad`** *`as xs:string`*`,` **`StringLength`** *`as xs:integer`*`,` **`PadCharacter`** *`as xs:string`***`) as xs:string`** **XP3.1** **XQ3.1**

The `PadCharacter` argument is a single character. It is padded to the right of the string to increase the number of characters in `StringToPad` so that this number equals the integer value of the `StringLength` argument. The `StringLength` argument can have any integer value (positive or negative), but padding will occur only if the value of `StringLength` is greater than the number of characters in `StringToPad`. If `StringToPad` has more characters than the value of `StringLength`, then `StringToPad` is left unchanged.

⊟ *Examples*

- `pad-string-right('AP', 1, 'Z')` returns `'AP'`
- `pad-string-right('AP', 2, 'Z')` returns `'AP'`
- `pad-string-right('AP', 3, 'Z')` returns `'APZ'`
- `pad-string-right('AP', 4, 'Z')` returns `'APZZ'`
- `pad-string-right('AP', -3, 'Z')` returns `'AP'`
- `pad-string-right('AP', 3, 'YZ')` returns a pad-character-too-long error

▼ repeat-string [altova:]

**`repeat-string(InputString`** *`as xs:string`*`,` **`Repeats`** *`as xs:integer`***`) as xs:string`** **XP2** **XQ1** **XP3.1** **XQ3.1**

Generates a string that is composed of the first `InputString` argument repeated `Repeats` number of times.

⊟ *Examples*

- `repeat-string("Altova #", 3)` returns "`Altova #Altova #Altova #`"

### substring-after-last [altova:]

**`substring-after-last(MainString`** *`as xs:string`*, **`CheckString`** *`as xs:string`***`)`** **`as xs:string`**
**XP3.1** **XQ3.1**
If `CheckString` is found in `MainString`, then the substring that occurs after `CheckString` in `MainString` is returned. If `CheckString` is not found in `MainString`, then the empty string is returned. If `CheckString` is an empty string, then `MainString` is returned in its entirety. If there is more than one occurrence of `CheckString` in `MainString`, then the substring after the last occurrence of `CheckString` is returned.

*Examples*

- `substring-after-last('ABCDEFGH', 'B')` returns `'CDEFGH'`
- `substring-after-last('ABCDEFGH', 'BC')` returns `'DEFGH'`
- `substring-after-last('ABCDEFGH', 'BD')` returns `''`
- `substring-after-last('ABCDEFGH', 'Z')` returns `''`
- `substring-after-last('ABCDEFGH', '')` returns `'ABCDEFGH'`
- `substring-after-last('ABCD-ABCD', 'B')` returns `'CD'`
- `substring-after-last('ABCD-ABCD-ABCD', 'BCD')` returns `''`

### substring-before-last [altova:]

**`substring-before-last(MainString`** *`as xs:string`*, **`CheckString`** *`as xs:string`***`)`** **`as xs:string`**
**XP3.1** **XQ3.1**
If `CheckString` is found in `MainString`, then the substring that occurs before `CheckString` in `MainString` is returned. If `CheckString` is not found in `MainString`, or if `CheckString` is an empty string, then the empty string is returned. If there is more than one occurrence of `CheckString` in `MainString`, then the substring before the last occurrence of `CheckString` is returned.

*Examples*

- `substring-before-last('ABCDEFGH', 'B')` returns `'A'`
- `substring-before-last('ABCDEFGH', 'BC')` returns `'A'`
- `substring-before-last('ABCDEFGH', 'BD')` returns `''`
- `substring-before-last('ABCDEFGH', 'Z')` returns `''`
- `substring-before-last('ABCDEFGH', '')` returns `''`
- `substring-before-last('ABCD-ABCD', 'B')` returns `'ABCD-A'`
- `substring-before-last('ABCD-ABCD-ABCD', 'ABCD')` returns `'ABCD-ABCD-'`

### substring-pos [altova:]

**`substring-pos(StringToCheck`** *`as xs:string`*, **`StringToFind`** *`as xs:string`***`)`** **`as xs:integer`** **XP3.1** **XQ3.1**
Returns the character position of the first occurrence of `StringToFind` in the string `StringToCheck`. The character position is returned as an integer. The first character of `StringToCheck` has the position `1`. If `StringToFind` does not occur within `StringToCheck`, the integer `0` is returned. To check for the second or a later occurrence of `StringToCheck`, use the next signature of this function.

*Examples*

- `substring-pos('Altova', 'to')` returns `3`

- **substring-pos**('Altova', 'tov') returns 3
- **substring-pos**('Altova', 'tv') returns 0
- **substring-pos**('AltovaAltova', 'to') returns 3

**substring-pos(StringToCheck** *as xs:string*, **StringToFind** *as xs:string*, **Integer** *as xs:integer*) **as xs:integer** **XP3.1** **XQ3.1**
Returns the character position of `StringToFind` in the string, `StringToCheck`. The search for `StringToFind` starts from the character position given by the `Integer` argument; the character substring before this position is not searched. The returned integer, however, is the position of the found string within the *entire* string, `StringToCheck`. This signature is useful for finding the second or a later position of a string that occurs multiple times with the `StringToCheck`. If `StringToFind` does not occur within `StringToCheck`, the integer `0` is returned.

*Examples*

- **substring-pos**('Altova', 'to', 1) returns 3
- **substring-pos**('Altova', 'to', 3) returns 3
- **substring-pos**('Altova', 'to', 4) returns 0
- **substring-pos**('Altova-Altova', 'to', 0) returns 3
- **substring-pos**('Altova-Altova', 'to', 4) returns 10

### trim-string [altova:]

**trim-string(InputString** *as xs:string*) **as xs:string** **XP3.1** **XQ3.1**
This function takes an `xs:string` argument, removes any leading and trailing whitespace, and returns a "trimmed" `xs:string`.

*Examples*

- **trim-string**(`"   Hello World   "`) returns `"Hello World"`
- **trim-string**(`"Hello World   "`) returns `"Hello World"`
- **trim-string**(`"   Hello World"`) returns `"Hello World"`
- **trim-string**(`"Hello World"`) returns `"Hello World"`
- **trim-string**(`"Hello   World"`) returns `"Hello   World"`

### trim-string-left [altova:]

**trim-string-left(InputString** *as xs:string*) **as xs:string** **XP3.1** **XQ3.1**
This function takes an `xs:string` argument, removes any leading whitespace, and returns a left-trimmed `xs:string`.

*Examples*

- **trim-string-left**(`"   Hello World   "`) returns `"Hello World   "`
- **trim-string-left**(`"Hello World   "`) returns `"Hello World   "`
- **trim-string-left**(`"   Hello World"`) returns `"Hello World"`
- **trim-string-left**(`"Hello World"`) returns `"Hello World"`
- **trim-string-left**(`"Hello   World"`) returns `"Hello   World"`

### trim-string-right [altova:]

**trim-string-right(InputString** *as xs:string*) **as xs:string** **XP3.1** **XQ3.1**

This function takes an `xs:string` argument, removes any trailing whitespace, and returns a right-trimmed `xs:string`.

⊟ *Examples*

- **`trim-string-right`**`("   Hello World   "))` returns `"   Hello World"`
- **`trim-string-right`**`("Hello World   "))` returns `"Hello World"`
- **`trim-string-right`**`("   Hello World"))` returns `"   Hello World"`
- **`trim-string-right`**`("Hello World"))` returns `"Hello World"`
- **`trim-string-right`**`("Hello   World"))` returns `"Hello   World"`

## 30.1.1.8 XPath/XQuery Functions: Miscellaneous

These miscellaneous extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of **`[altova:]`**. For example: **`add-years-to-date [altova:]`**.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

| | |
|---|---|
| *XPath functions (used in XPath expressions in XSLT):* | XP1 XP2 XP3.1 |
| *XSLT functions (used in XPath expressions in XSLT):* | XSLT1 XSLT2 XSLT3 |
| *XQuery functions (used in XQuery expressions in XQuery):* | XQ1 XQ3.1 |

▼ decode-string [altova:]

**`decode-string(Input`** *`as xs:base64Binary`***`) as xs:string`** XP3.1 XQ3.1
**`decode-string(Input`** *`as xs:base64Binary`*, **`Encoding`** *`as xs:string`***`) as xs:string`** XP3.1 XQ3.1
Decodes the submitted base64Binary input to a string using the specified encoding. If no encoding is specified, then the UTF-8 encoding is used. The following encodings are supported: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

⊟ *Examples*

- **`decode-string`**`($XML1/MailData/Meta/b64B)` returns the base64Binary input as a UTF-8 encoded string
- **`decode-string`**`($XML1/MailData/Meta/b64B, "UTF-8")` returns the base64Binary input as a UTF-8-encoded string
- **`decode-string`**`($XML1/MailData/Meta/b64B, "ISO-8859-1")` returns the base64Binary input

as an ISO-8859-1-encoded string

▼ encode-string [altova:]

**`encode-string(InputString`** *`as xs:string`***`) as xs:base64Binaryinteger`** **XP3.1** **XQ3.1**
**`encode-string(InputString`** *`as xs:string`***`, Encoding`** *`as xs:string`***`) as xs:base64Binaryinteger`** **XP3.1** **XQ3.1**
Encodes the submitted string using, if one is given, the specified encoding. If no encoding is given, then the UTF-8 encoding is used. The encoded string is converted to base64Binary characters, and the converted base64Binary value is returned. Initially, UTF-8 encoding is supported, and support will be extended to the following encodings: `US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4`

⊟ *Examples*

- `encode-string("Altova")` returns the base64Binary equivalent of the UTF-8 encoded string `"Altova"`
- `encode-string("Altova", "UTF-8")` returns the base64Binary equivalent of the UTF-8 encoded string `"Altova"`

▼ generate-guid [altova:]

**`generate-guid() as xs:string`** **XP2** **XQ1** **XP3.1** **XQ3.1**
Generates a unique string GUID string.
⊟ *Examples*

- `generate-guid()` returns (for example) `85F971DA-17F3-4E4E-994E-99137873ACCD`

▼ high-res-timer [altova:]

**`high-res-timer() as xs:double`** **XP3.1** **XQ3.1**
Returns a system high-resolution timer value in seconds. A high-resolution timer, when present on a system, enables high precision time measurements when these are required (for example, in animations and for determining precise code-execution time). This function provides the resolution of the system's high-res timer.
⊟ *Examples*

- `high-res-timer()` returns something like `'1.16766146154566E6'`

▼ parse-html [altova:]

**`parse-html(HTMLText`** *`as xs:string`***`) as node()`** **XP3.1** **XQ3.1**
The `HTMLText` argument is a string that contains the text of an HTML document. The function creates an HTML tree from the string. The submitted string may or may not contain the HTML element. In either case, the root element of the tree is an element named **`HTML`**. It is best to make sure that the HTML code in the submitted string is valid HTML.
⊟ *Examples*

- `parse-html("<html><head/><body><h1>Header</h1></body></html>")` creates an HTML tree from the submitted string

▼ sleep[altova:]

**sleep(Millisecs** *as xs:integer***)** **as empty-sequence()** XP2 XQ1 XP3.1 XQ3.1

Suspends execution of the current operation for the number of milliseconds given by the `Millisecs` argument.

⊟ *Examples*

- `sleep(1000)` suspends execution of the current operation for 1000 milliseconds.

[ Top 1764 ]

# 30.2 License Information

This section contains information about:

- the distribution of this software product
- software activation and license metering
- the license agreement governing the use of this product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

To view the terms of any Altova license, go to the Altova Legal Information page at the Altova website.

## 30.2.1 Electronic Software Distribution

This product is available through electronic software distribution, a distribution method that provides the following unique benefits:

- You can evaluate the software free-of-charge for 30 days before making a purchasing decision. *(Note: Altova MobileTogether Designer is licensed free of charge.)*
- Once you decide to buy the software, you can place your order online at the Altova website and get a fully licensed product within minutes.
- When you place an online order, you always get the latest version of our software.
- The product package includes an onscreen help system that can be accessed from within the application interface. The latest version of the user manual is available at www.altova.com in (i) HTML format for online browsing, and (ii) PDF format for download (and to print if you prefer to have the documentation on paper).

### Helping Others within Your Organization to Evaluate the Software

If you wish to distribute the evaluation version within your company network, or if you plan to use it on a PC that is not connected to the Internet, you may distribute only the installer file, provided that this file is not modified in any way. Any person who accesses the software installer that you have provided must request their own 30-day evaluation license key code and after expiration of their evaluation period, must also purchase a license in order to be able to continue using the product.

## 30.2.2 Software Activation and License Metering

As part of Altova's Software Activation, the software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration, use, or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the software and to improve customer service. Activation is based on the exchange of license related data such as operating system, IP address, date/time, software version, and computer name, along with other information between your computer and an Altova license server.

Your Altova product has a built-in license metering module that further helps you avoid any unintentional violation of the End User License Agreement. Your product is licensed either as a single-user or multi-user installation, and the license-metering module makes sure that no more than the licensed number of users use the application concurrently.

This license-metering technology uses your local area network (LAN) to communicate between instances of the application running on different computers.

## Single license

When the application starts up, as part of the license metering process, the software sends a short broadcast datagram to find any other instance of the product running on another computer in the same network segment. If it doesn't get any response, it will open a port for listening to other instances of the application.

## Multi-user license

If more than one instance of the application is used within the same LAN, these instances will briefly communicate with each other on startup. These instances exchange key-codes in order to help you to better determine that the number of concurrent licenses purchased is not accidentally violated. This is the same kind of license metering technology that is common in the Unix world and with a number of database development tools. It allows Altova customers to purchase reasonably-priced concurrent-use multi-user licenses.

We have also designed the applications so that they send few and small network packets so as to not put a burden on your network. The TCP/IP ports (2799) used by your Altova product are officially registered with the IANA (*see the IANA Service Name Registry for details*) and our license-metering module is tested and proven technology.

If you are using a firewall, you may notice communications on port 2799 between the computers that are running Altova products. You are, of course, free to block such traffic between different groups in your organization, as long as you can ensure by other means, that your license agreement is not violated.

## Note about certificates

Your Altova application contacts the Altova licensing server (`link.altova.com`) via HTTPS. For this communication, Altova uses a registered SSL certificate. If this certificate is replaced (for example, by your IT department or an external agency), then your Altova application will warn you about the connection being insecure. You could use the replacement certificate to start your Altova application, but you would be doing this at your own risk. If you see a *Non-secure connection* warning message, check the origin of the certificate and consult your IT team (who would be able to decide whether the interception and replacement of the Altova certificate should continue or not).

If your organization needs to use its own certificate (for example, to monitor communication to and from client machines), then we recommend that you install Altova's free license management software, Altova LicenseServer, on your network. Under this setup, client machines can continue to use your organization's certificates, while Altova LicenseServer can be allowed to use the Altova certificate for communication with Altova.

### 30.2.3 Altova MobileTogether Designer End User License Agreement

- The Altova MobileTogether Designer End-User License Agreement is available here: https://www.altova.com/legal/mobiletogether-eula
- Altova's Privacy Policy is available here: https://www.altova.com/privacy

# Index