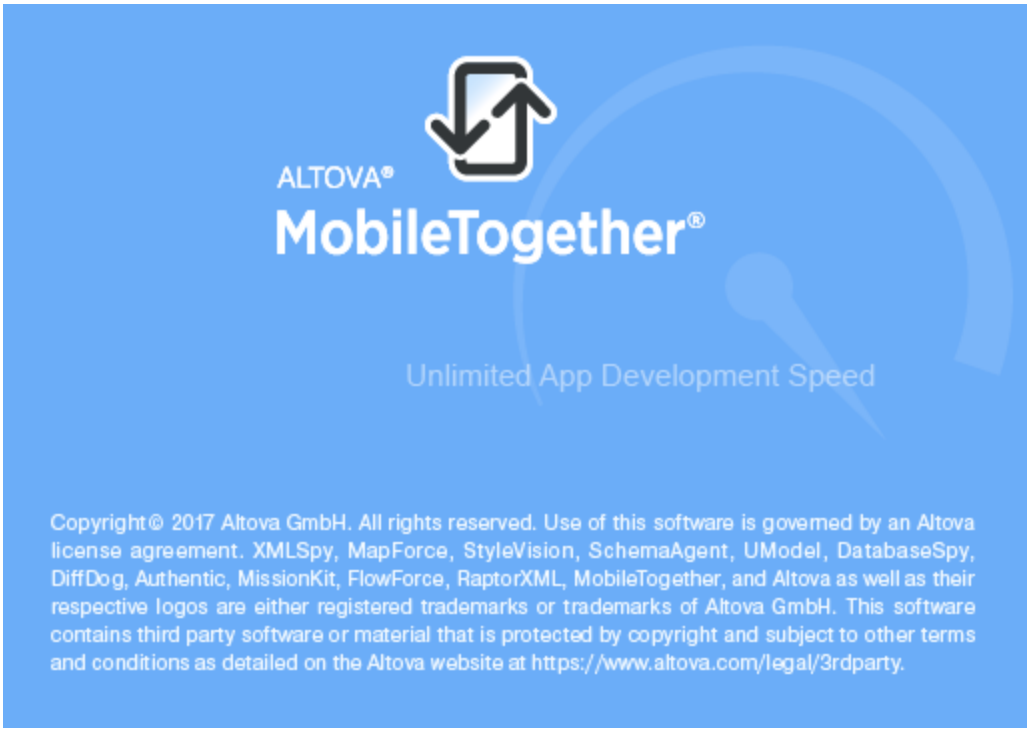
User and Reference Manual



Altova MobileTogether Designer
User & Reference Manual

Published: 2018

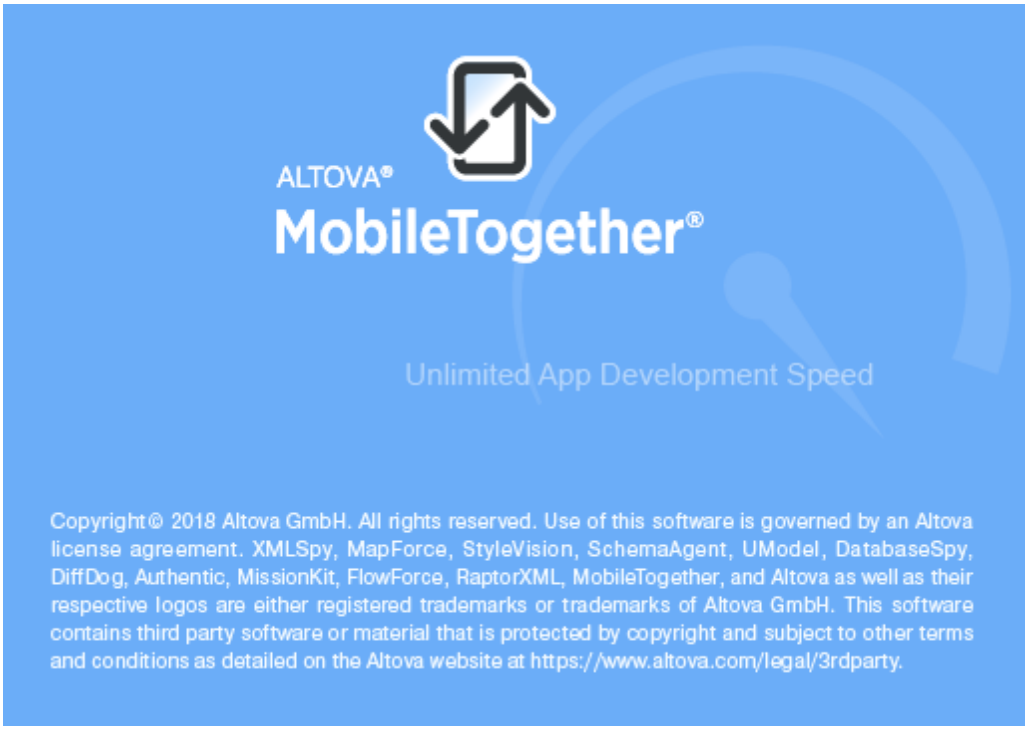
Table of Contents

Chapter 1

Altova MobileTogether Designer

1 Altova 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.



MobileTogether Designer video demos

At the Altova website you can view **video demos** that show how to use MobileTogether Designer to build a variety of MobileTogether solutions. These videos provide a fast introduction to the powerful features of MobileTogether Designer.

This documentation

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

- Introduction
- Tutorials
- User Interface
- Project

Current version: 4.1

Last updated: 26 February 2018

Chapter 2

New Features

2 New Features

Features that are new in MobileTogether Designer **Version 4.1** are listed below. These are followed by lists containing the new features of previous releases of Version 4.

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** of MobileTogether Designer.
- How to create a server service in MobileTogether Designer is described in the section Server Services.
- A `$MT_SERVICE` page source is automatically created when a service design is created. It holds run time data about service triggers.
- The `$MT_SERVICE` page source can be manually filled to simulate run time data about service triggers.

Rich Text

- A new Rich Text control 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.
- For each project (design), you can define multiple Rich Text style sheets via the Rich Text Style Sheets dialog. Any one of these style sheets can be assigned to a Rich Text control 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.

Actions

- The Go to Subpage 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 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 action saves information about the device's calendars and calendar events to the `$MT_CALENDAR` page source. It also enables events to be written to a calendar on the device. For simulations, the calendar of Microsoft Outlook or an XML file can be used.
- A Replace Node(s) 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 is enhanced to enable users to select multiple options (via the control's **`Multi Select`** property).
- Controls that have a **`Text Size`** property now also have a **`Text Size Auto-Fit`** property, which enables text to be 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 Message pane by using the Page menu command **List Text Size Auto-Fit Groups**.
- 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** 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, Check Boxes, Combo Boxes, Dates, Edit Fields, Radio Buttons, Switch controls, Time controls.
- Controls that have an **`OnClicked`** event (Buttons, Charts, Images, and Labels) 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: (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 **`generate-guid`** generates a unique GUID string that can be used as an id.

Miscellaneous

- Enforce Light Theme: In the Project Properties 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 and Access Calendar actions. This is done by selecting the corresponding items in the Options dialog.

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.

- The Send Push Notification 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** 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 action of the sending solution. While definitions of the buttons for non-iOS devices are made directly in the Send Push Notification action, for iOS devices, the buttons are defined in the receiving solution by using the **Project | iOS Push Notification Button Sets** command.
- An external PN key is a text string that is used to identify a mobile device. The Register Ext PN-Key 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, is also available.
- A PN topic is a text string that names a topic. The Register PN-Topics 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, 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.
- A MobileTogether solution that uses PNs can be compiled into an AppStore App. A few additional steps are required to compile AppStore Apps. These steps are described in Push Notifications in AppStore Apps.

Embedded Webpage Solutions

- A new Embedded Webpage Solutions 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) enables embedded webpage solutions to be integrated into existing systems.
- The **OnEmbeddedMessage** event is triggered when a solution's workflow on the server receives a message from the embedded solution.
- The **$MT_EMBEDDEDMESSAGE** JSON page source (structure and data) is created when the **OnEmbeddedMessage** event is triggered.
- The Load from String action parses a string and generates a (JSON/XML) page source from it.
- The Save to String action serializes a selected (JSON/XML) page source, and saves the serialized string to a specified location.
- The Embedded Message Back action sends a serialized JSON string as a `message` event to the IFrame that loaded the current solution.

New actions

- The MapForce Transfer 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 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 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 action parses a string and generates a (JSON/XML) page source from it.
- The Save to String action serializes a selected (JSON/XML) page source, and saves the serialized string to a specified location.
- The Embedded Message Back action sends a serialized JSON string as a `message` event to the IFrame that loaded the current solution.
- The Send Push Notification action defines the various parameters of the push notification that is to be sent.
- The (Un)Register Ext PN-Key action registers a text string as the external Push Notification key of a solution on that mobile device. See the section Push Notifications for more information.
- The (Un)Register PN-Topics action registers a device to receive Push Notifications about one or more selected topics. See the section Push Notifications 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. The tools created in this way are accessed via commands in the **Tools | User-Defined Tools** menu.
- A new Table menu 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** 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** command is clicked.
- Users can swipe right/left to horizontally scroll tables that are broader than the viewport.
- The simulator's menu provides options to simulate the availability of the following mobile device functionality: (i) the camera app, (ii) the gallery, (iii) the microphone, (iv) NFC, (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) that relate to specific actions of specific events can be suppressed or enabled as needed.
- Page source data can be automatically reset when the solution exits a page. This is done with the **Reset Data** command, which is available in the context menu of data sources.
- When saving (any type of) files, optionally, a default file extension can be specified; this extension will be used if none is specified with the file name.
- During simulations, you can copy the XPath locator expression of any page source node to the clipboard.

2.1 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 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.
- NFC-related events to trigger actions: `OnPushNdefMessageCompleted` and `OnNFCTagDiscovered`.
- A new MobileTogether extension function to check whether NFC has been started: `mt-nfc-started`. Plus functions to convert text and Base64 to/from hexBinary (since NFC message payloads are encoded in hexBinary).
- NFC sample files enable the simulation of NFC-tag discovery.

Text to Speech

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

Miscellaneous new actions

- A new Read Contacts action to store the contacts of the device's address book in a data source tree.
- A new Get File Info action to store the file information of a specified file (such as size, creation date, etc) in a data source tree.
- A new Wait Cursor 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 and Let User Choose Time actions enable the end user's selection of date and time, respectively, to be saved to page source nodes.
- A View Image 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 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 in the design.
- The Audio action 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 now contains a *Reply To* setting. This enables emails sent via MobileTogether Server to have both "pseudo" and real sender-addresses.
- The Reset action now enables all data sources, including the `$PERSISTENT` tree (for persistent data on the client), to be reset.

- The Show Geolocation 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 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 to check whether geolocation tracking has been started: `mt-geolocation-started`.
- The target pages of the Go to Page and Go to Subpage actions can additionally be set via XPath expressions.
- In message boxes containing custom buttons, 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.

New features of tables

- Tables can have dynamic columns, 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, **`MT_TableColumnContext`**, has been added. It provides the context node of the current column during the generation of tables. See the section Dynamic Columns for a description of usage.
- The number of rows that can be loaded in scrollable tables can be set with the **`Row Group Chunk Size`** table property.

Enhancements for controls

- Two new control properties 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 are available: *Import, Export, Calendar,* and *Time*.
- Images that have been embedded 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, 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 can be pasted as XPath into the Styles & Properties Pane.
- Additional toolbar icons in the Style Sheets dialog 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 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 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 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 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** (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** switches the padding of tables on iOS devices on or off.
- The Automated Testing feature enables you to compare two test runs to detect differences in the design, page source data, and the solution environment.

2.2 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, Load/Save Image, and Load/Save Binary File.
- Video controls enable videos to be played on a page. The control's properties and video-related MobileTogether extension functions enable the playback and the control to be customized. For an overview of video features, see the section Audio, Video.
- A Video action 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.
- An Audio action 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.
- The **`$MT_AudioChannel`** global variable gives the number of the audio channel that triggered the action.
- An Audio Recording action enables audio to be recorded to a file on the client device. For an overview of audio features, see the section Audio, Video.
- New MobileTogether extension functions for providing information about audio and video files, and about actions related to audio and video: `mt-audio-get-current-position`, `mt-audio-get-duration`, `mt-audio-is-playing`, `mt-audio-is-recording`, `mt-video-get-current-position`, `mt-video-get-duration`, `mt-video-height`, `mt-audio-is-playing`, and `mt-video-width`.
- New MobileTogether extension functions for providing information about the last client file that was used: `mt-last-file-path`, `mt-extract-file-extension`, and `mt-extract-file-name`.
- New button icons related to the audio/video features can be selected via the `Button Look` property.
- New global variables **`$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 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 action can send text-file attachments, in addition to XML files and binary files.
- The Simulator can be set to simulate the availability of a LAN connection. This adds to the number of connection types that can be simulated, which now are: mobile network, WiFi, and LAN. A related MobileTogether extension function has been introduced: (i) `mt-connected-via-lan`.
- Table headers and footers can be added to dynamic tables via the context menus of tables.

Version 2.1

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

- Data files, such as XML and image files, can be loaded from client devices and saved to client devices.
- Two new controls are available: Vertical Line and Horizontal Slider.
- The following actions have been introduced:
  - Share
  - Cancel Action Execution
  - User Cancel Behavior
  - Restart/Stop Page Timer
  - Delete File/Folder
  - DB Bulk Insert Into
  - Let
  - Try/Catch
  - Throw
  - Return
- Action Group Results: A Return action in an Action Group generates an Action Group Result, which can be used as the value of a variable defined in a Let action.
- Action Groups can take parameters. Additionally, an Action Group itself can be set as as the value of a parameter.
- The Close Subpage action has been extended to return a value that can be used as the value of a variable defined in a Let action.
- The Show Geolocation (on map) action has been enhanced to show the routes between two locations.
- Emails sent from clients can be in HTML format.
- The precision of timers used for the page refresh event has been increased to milliseconds.
- XPath definitions of the following properties: `Keyboard` (of the Edit Field control), `Horizontal Alignment`, and `Vertical Alignment`.
- The width of controls and table columns can be set in pixels.
- Tables: A whole table or a part of a table can be designed to be scrollable. Scrollable tables can be specified to fill the screen height.
- Tables: Separate visibility settings for spanned columns and rows.
- Tables: Background colors can be assigned to individual rows and columns (in addition to cells).
- Tables: Nested tables can be assigned horizontal-alignment and vertical-alignment property values.
- The `Keyboard` property of the Edit Field control 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 has additional predefined looks (specified via the `Button Look` property), including transparent buttons.
- New MobileTogether extension functions: (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 **mt-font-height** function.
- When saving to a DB, columns can be filtered separately depending on whether data is being updated or inserted.
- The `Show Page Title Bar` page property enables the page's title bar to displayed or hidden.
- User-generated AppStore Apps: The app's UI language can be selected from among English, German, French, Spanish, and Japanese.
- Duplication of custom localization strings.

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 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*, 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 action.
- A Signature Field 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 occur.
- Simulations have been enhanced to better emulate actions defined in the design. For example, server connection errors are simulated by an option to prevent server access.
- JSON data sources can be used as page sources.
- Page data can be accessed and saved via REST requests. Such data can be used in page sources, and can also be accessed or saved via page source actions.
- REST requests support OAuth authorization. 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 dialog. Furthermore, settings can be imported into the active document from other open MobileTogether Designer documents.
- Page data can be accessed and saved via SOAP requests. Such data can be used in page sources and page source actions.
- New actions: Execute SOAP Request, Execute REST Request.
- The data retention option for page sources offers considerable flexibility about whether data is stored on the client or server.
- A page event, `OnServerConnectionError`, has been added.
- Two dynamic, local variables have been added: **`MT_HTTPExecute_Result`** and **`MT_ServerConnectionErrorLocation`**.
- Commands to list all files, directories, and external data sources that are used in the project.
- Cells of Repeating Tables and Dynamic Tables are associated with page source nodes via XPath expressions, and were previously read-only. The content of such cells are now editable.

2.3 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 action enables emails to be sent during the execution of a solution.
- The MobileTogether extension function **mt-email-attachment** creates text and image attachments for emails that are sent with the Send Email To action.
- Links can be placed in the body of emails 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.
- Hyperlinks that target solutions pass their URL query parameters to the targeted solution. These parameters can be stored in the **$MT_InputParameters** global variable, from where they can be referenced.
- Three link-related MobileTogether extension functions have been added: **mt-run-solution-url**, **mt-run-solution-url-parameters**, and **mt-html-anchor**.
- A powerful Loop 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, and Update Display.
- A new radio button control has been introduced.
- The strings of a solution automatically appear in the language of a mobile device if the solution has been localized in that language. In this release, the default and localization strings can be exported/imported between the project and separate XML files 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** function in an XPath expression in the Edit XPath/XQuery Expression dialog, all the custom strings defined in the project are displayed in a popup. The value of the string in the simulation language currently selected in MobileTogether Designer is also displayed.
- A new function, **mt-localized-string-name**, returns the control name or string name of the submitted (localized) string.
- The button 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 control has the following new properties: `Line Style`, `Margin Top`, `Margin Bottom`.
- The width of all controls 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, Charts, Images, and Labels.
- The Insert Node(s) and Append Node(s) actions have an option to remove the inserted/appended node/s from their original locations in project data sources.

- Keyboard shortcuts for adding actions to the definition of an event.
- Each control in the design 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.
- An external CSS file can be used to store additional CSS styles.
- A new dialog for a project's browser settings collects the settings that define the behavior of the browser in the client mobile device.
- Custom fonts can be embedded in a design.
- Enhancements to the XPath/XQuery Expression dialog include interactive functions-and-operators information in popups, information about global variables and custom strings.
- User-Defined XPath/XQuery Functions can be ordered in the ascending/descending/dialog order of function names.
- Updating server settings on client devices.

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 can be defined for events. Additionally, Altova XPath extension functions for manipulating geolocation data can be used in the design's XPath expressions. Geolocations can also be set for designer and server simulations 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 to create and check XQuery expressions.
- String localization (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.
- Specific headers can now be added to HTTP requests. 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 action.]
- Simulations have been enhanced for iOS7/8 rendering and for XML tree editing. 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 contains commands to show: (i) used global and page source variables; (ii) used user-defined XPath/XQuery functions; (iii) used action groups; (iv) as well as unused variables, functions, and action groups. This improves the maintenance and development of large, complex solutions.

Chapter 3

Introduction

3 Introduction

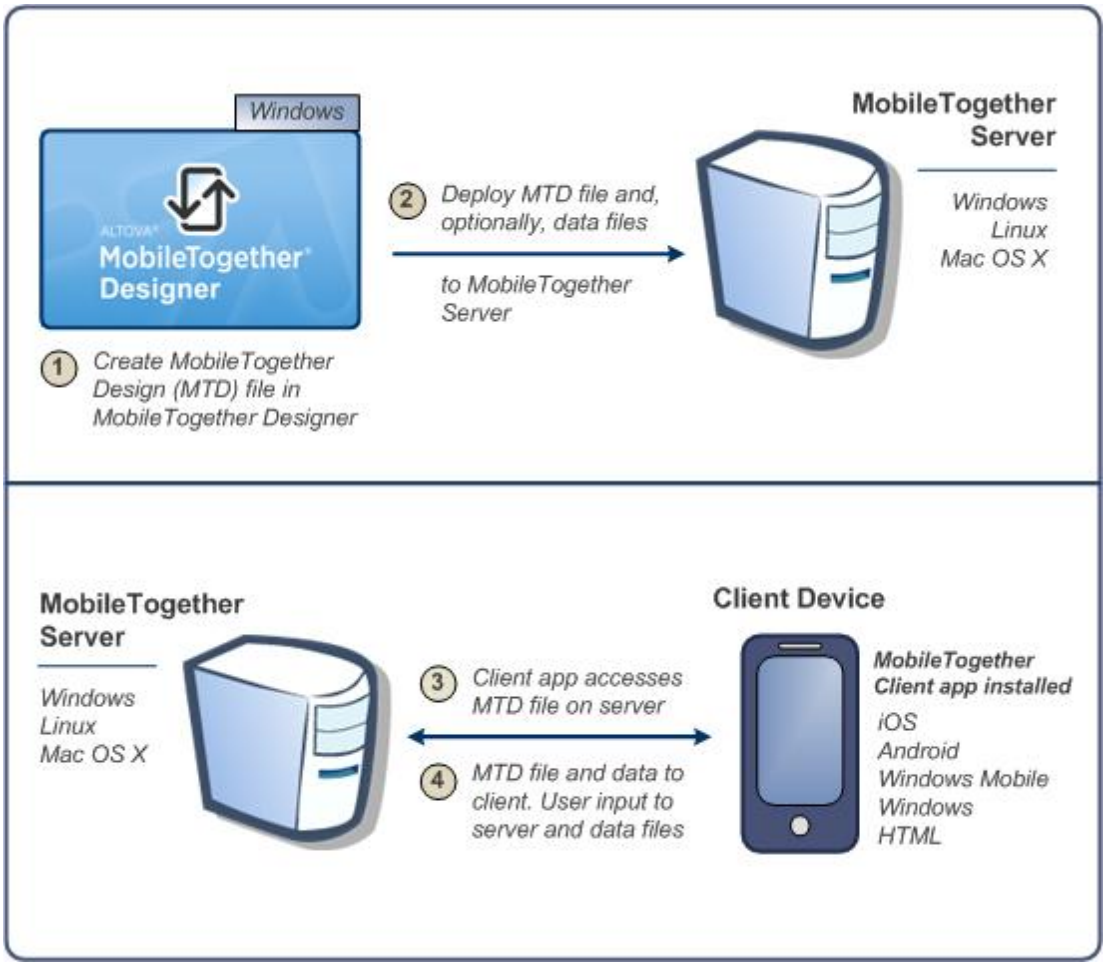
This section provides an overview of MobileTogether and MobileTogether Designer. It contains the following sections:

- MobileTogether Overview
- Terminology Q&A
- Design Steps
- XPath in MobileTogether

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.



System requirements

▼ MobileTogether Designer

Windows	Windows 7 SP1 with Platform Update, Windows 8,

	Windows 10
Windows Server	Windows Server 2008 R2 SP1 with Platform Update or newer

▼ MobileTogether Server

Windows	Windows 7 SP1 with Platform Update, Windows 8, Windows 10
Windows Server	Windows Server 2008 R2 SP1 with Platform Update or newer
Linux	• CentOS 6 or newer • RedHat 6 or newer • Debian 7 or newer • Ubuntu 12.04 or newer
(Mac) OS X, macOS	OS X10.10, 10.11, macOS 10.12 or newer

▼ MobileTogether Client

iOS	9 and higher for Apple mobile devices
Android	4.0 and higher for Android mobile devices
Windows RT, Metro	Windows 8.1, 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 format defined in the MTD file. End users can also update data files from their mobile client devices (via MobileTogether solutions).

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. 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 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 (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 to perform when an event of a control (control event, for short) is triggered.
- For each page, a set of data sources can be defined in the Page Sources Pane of that page.
- The content associated with a control can come from one (or more) of these data sources. Such data is accessed using the XPath or 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:* 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:* 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 data source. You can add 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** command. The file is created in memory and must be saved with the **File | Save** command to store it on disk. Define Project Properties and specify other project-related settings.

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 and each data 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 data source trees are addressed using XPath expressions. Client data input can also be saved back to the data sources if desired. See the section Page Sources 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. Each control has a set of properties (defined in the Styles & Properties Pane) and data (from the data source trees) 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 and Actions. 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 data 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. The sequence of top pages in the Pages Pane 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` action (of control or page events). Other page-related actions 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 within MobileTogether Designer. The simulation shows (in MobileTogether Designer itself) how the workflow will be executed on the client device. Select **Project | Simulate Workflow** or press **F5** to start the simulation. The Messages Pane 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 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 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 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 by 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, 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 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 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 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 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 ""
```

Chapter 4

Tutorials

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 take you through the basics, all the way up to deploying the MTD file and associated files on MobileTogether Server.
- The Simple Database tutorial shows how to create a DB-based design and how to load DB records on the basis of a user selection.
- The Database-And-Charts tutorial uses a finished design file to explain how to work with databases and how to create charts.
- The SubPages-And-Visibility 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 tutorial explains a number of intermediate-level concepts that will help you gain more confidence in using MobileTogether Designer.
- The SOAP Requests 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 tutorial shows how the Geolocation functionality can be used. Other features that are described include sharing and catching errors.
- The Scrollable Tables tutorial describes the key features of scrollable tables, and shows how such tables are created.

Finished design files, as well as the data sources and image files used in the tutorials, are available in the *(My) Documents* folder: `Altova\MobileTogetherDesigner4\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.

File paths in Windows 7, Windows 8, and Windows 10

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	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	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 data source, 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
- Set Up a Page
- Add a Page Data Source
- Format the Design
- Add a Control: Combo Box
- Add a Control: Image
- Define Control Actions
- Validate the Project
- Run a Simulation
- Deploy to Server

The tutorial files

The files for this tutorial are located in your *(My) Documents* 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 folder

Create a New Design

In this part, you will:

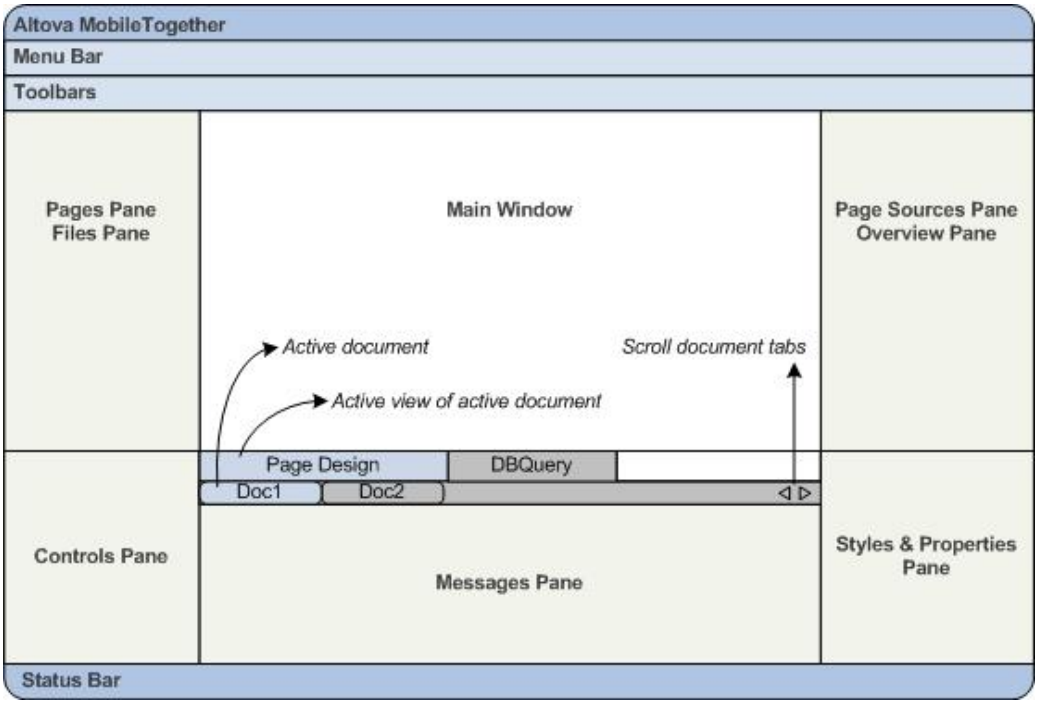
- Create a new MobileTogether design
- Save the design to file
- Set up the design's preview properties: the preview device and magnification level
- Familiarize yourself with the GUI of MobileTogether Designer

---

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, 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.

4. Familiarize yourself with the Main Window and its tabs, and with the different panes that are located around the main window. The user interface is described in the section, The User Interface.
5. Switch between the Page Design and DB Query tabs of the Main Window, and see how the contents of the panes change.

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. 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.

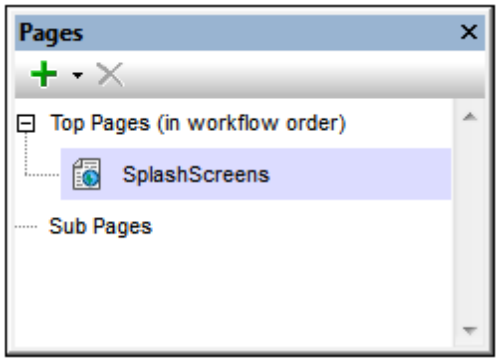
We are building a design that has one page. In this part, you will:

- 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 (*screenshot below*), double-click `New Page1` and rename it to **`SplashScreens`**.



2. From the Controls Pane, drag and drop the Label control 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 data sources. (No data source has been defined for the `SplashScreens` page as yet. This will be done in the next part.) 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 selected, the label's properties are displayed in the Styles & Properties Pane. If you like, modify any of the label's formatting properties available in this pane, such as the margin-bottom property to create empty space below the label.

Add a Page Data Source

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

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

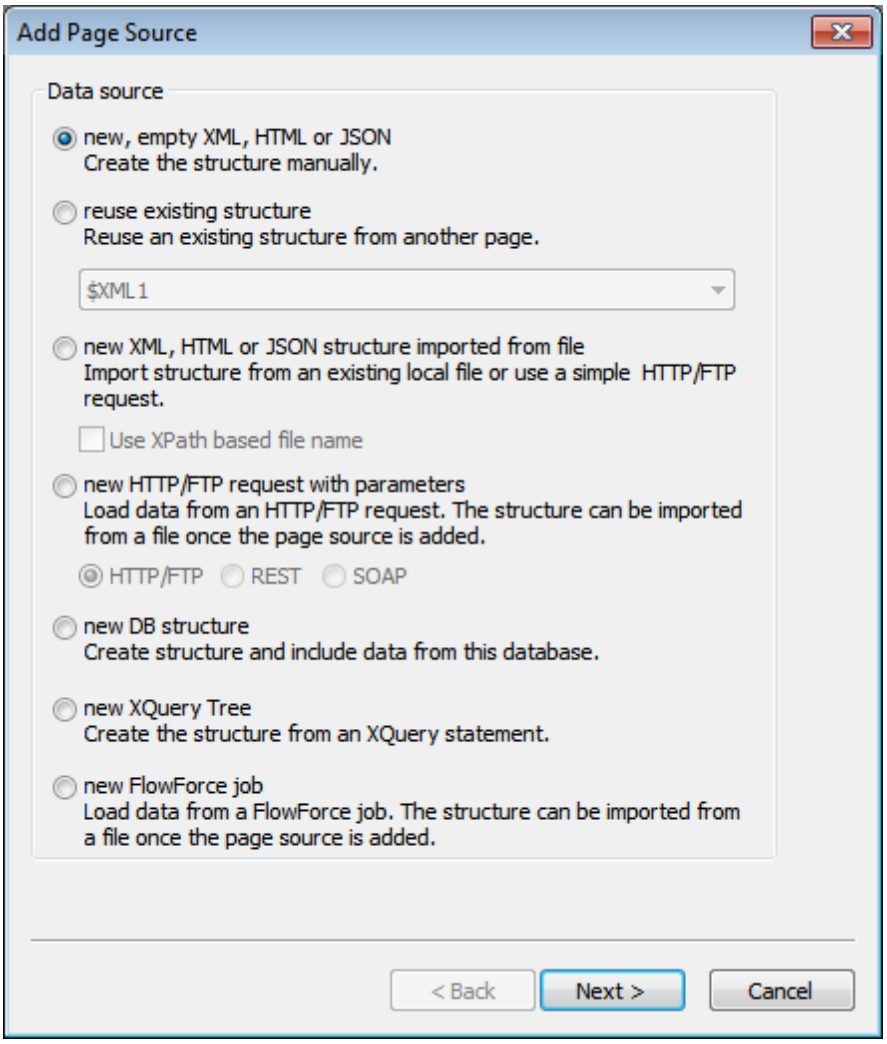
Buttons in this section

 **Add Source**

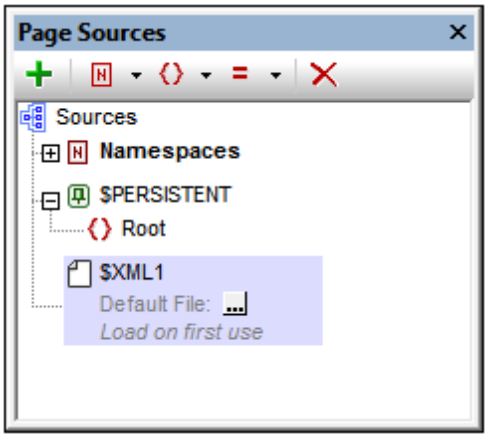
A page can have one or more data sources from where data can be obtained. Data sources are defined in the Page Sources Pane and can be specified to be read-only or editable. They can be external sources, or can be created directly in the Page Sources Pane and contain static data. The types of data sources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

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

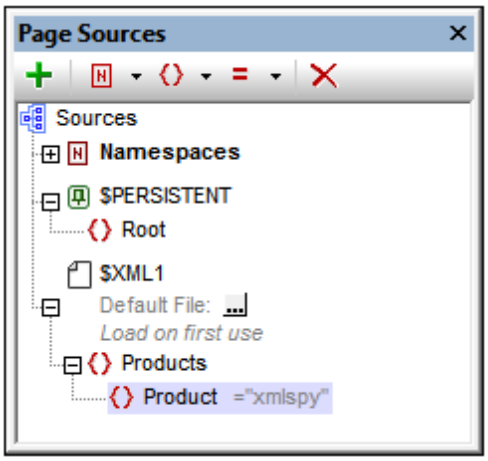
1. Click the **Add Source** button in the toolbar of the Page Sources Pane to display the Add Page Source dialog (*screenshot below*).



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 data source is created as editable, data in it can be modified.) A root node called `$XML1` is added to the Page Sources tree in the Page Sources Pane (*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 `$XML1`. A default file would provide the data that goes into the nodes of the data source's tree. The data provided by a default file can be overridden by data that is manually entered in a tree node. In our case, there is a single data node: **`$XML1/Products/Product`**, and it has the string **`xmlspy`** as its default content. The XML data structure is:

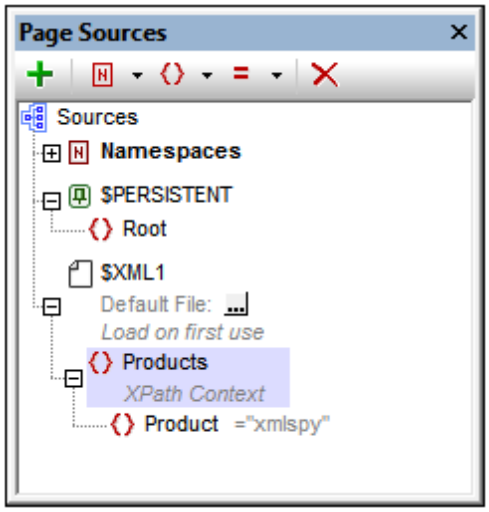
```
<Products>
   <Product>xmlspy</Product>
</Products>
```

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 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 data 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. For example: **$XML2/SomeRootElement/SomeOtherElement**.

Format the Design

The design page now has a name (SplashScreens), a display title (Altova Splash Screens), and a data source ($XML1). 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, drag the Table control 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, drag and drop the Label control 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, drag and drop the Horizontal Line control 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, 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.

5. Select the label control and set `Margin Top` and `Margin Bottom` properties in the Styles & Properties Pane.
6. Try copying different components to various parts of the design.

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 data 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.

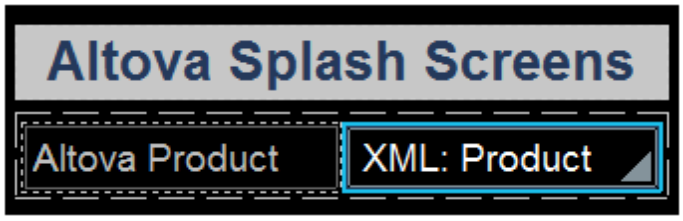
Buttons in this section

... **Additional Dialog**

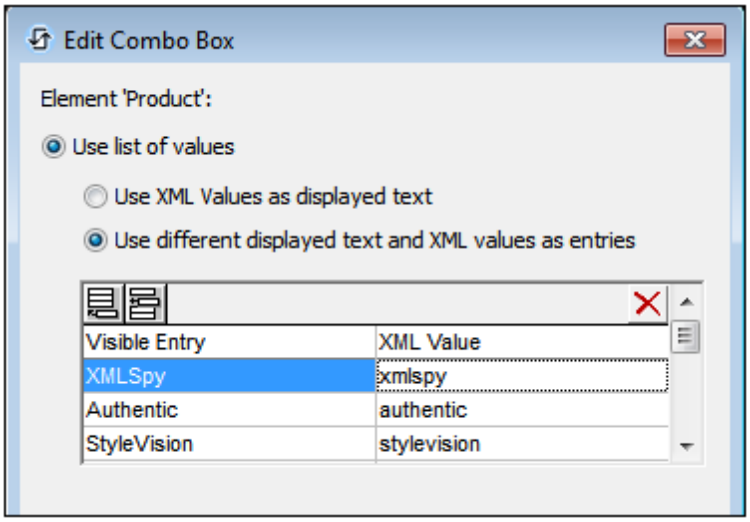
Add a combo box

Add the combo box and define its properties as follows:

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

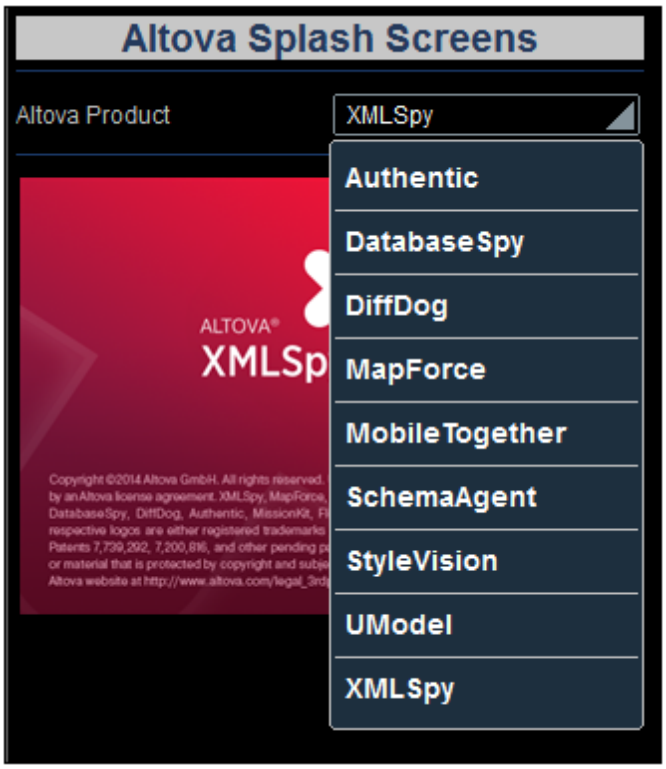


2. Drag the **Product** node from the Page Sources Pane, and drop it on to the combo box. This associates the **Product** node with the combo box, and creates what we call a source node (or page source link). When the end user selects an option from the combo box, the XML value of the selected option 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, click the **Additional Dialog** button of the *Combo Box Entry Values* property. The Edit Combo Box dialog (screenshot below) appears.



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, 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, 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 data source. Note that the default content of `Product` is `xmlspy` (defined when the node was created). 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.

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*). 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`, located in your *(My) Documents folder*. If you like you can save the images to another folder.

Buttons in this section

XPath

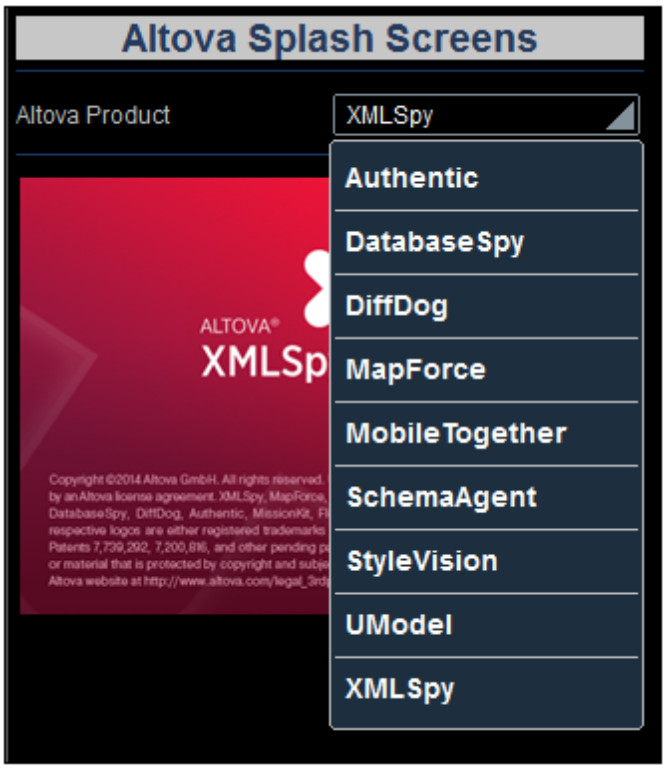
Add an image

Add the image and define its properties as follows:

1. From the Controls Pane, drag and drop the Image control below the table.
2. In the Styles & Properties Pane, name the image by setting its `Name` property to `Image: SplashScreen`.
3. Select the **`Image Source`** property, and click the **XPath** button in the toolbar of the Styles & Properties Pane. This displays the Edit XPath/XQuery Expression dialog.
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`**.

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.

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.

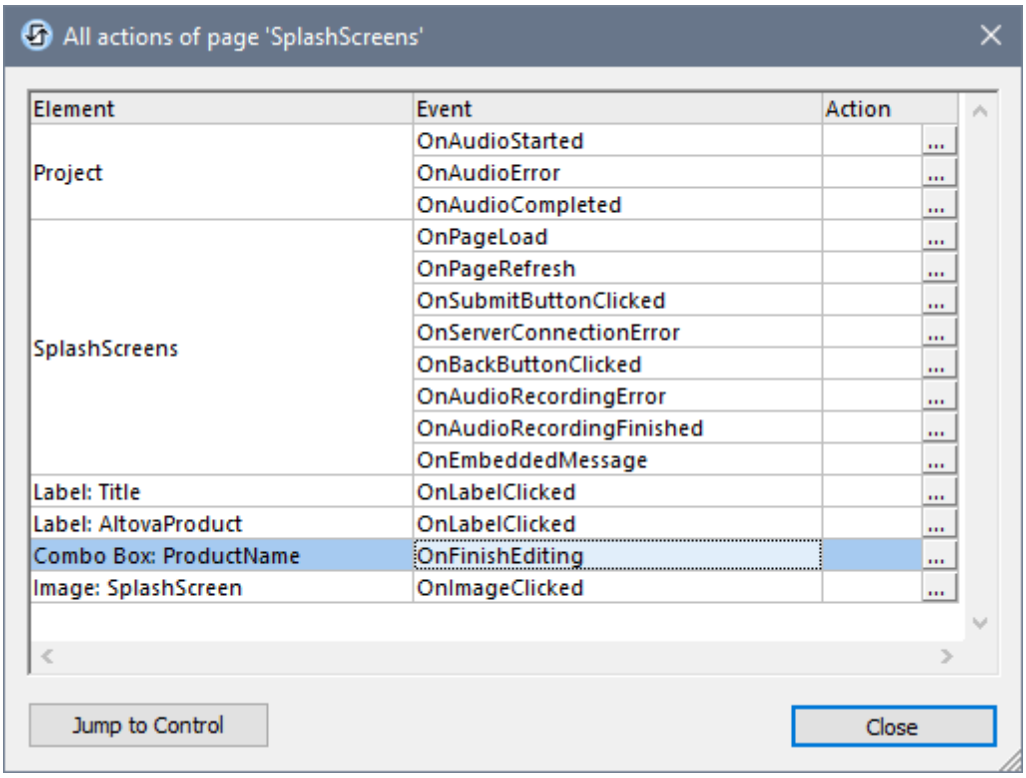
In this part, you will:

- 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.

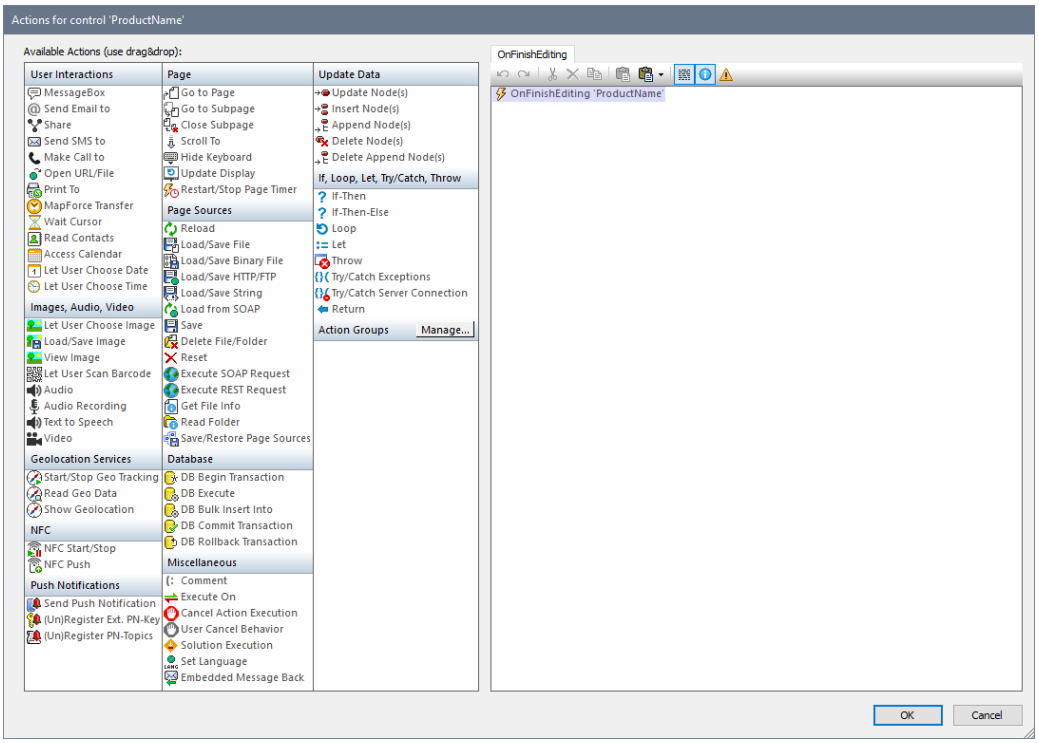


Define the Reload action on the combo box control

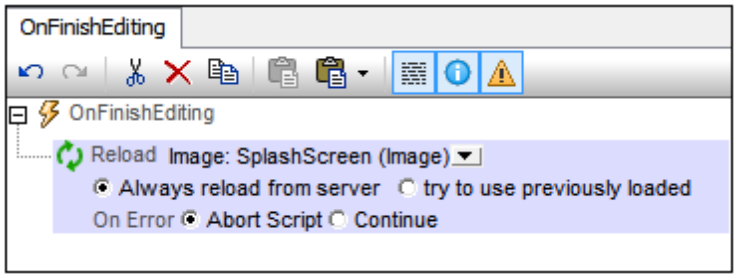
Defining a control action consists of two parts: (i) selecting the control event that triggers the

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.

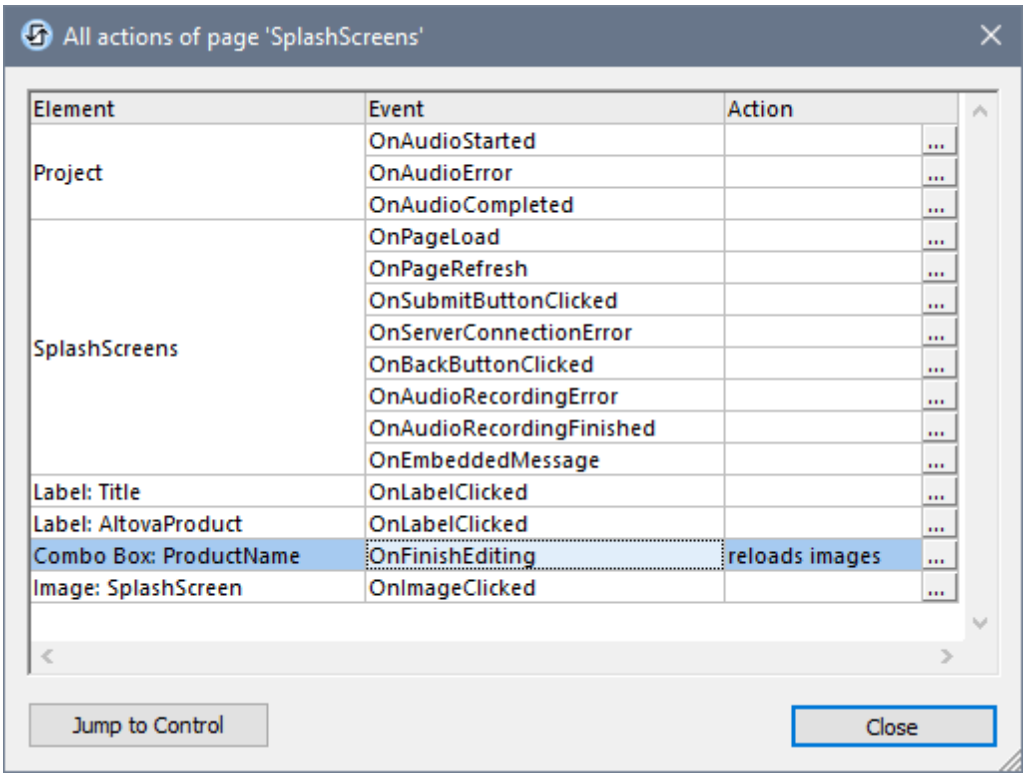
1. Right-click the **`XML:Product`** combo box and select **Control Actions on OnFinishEditing**. This opens the Control Actions dialog for the combo box (*screenshot below*). There is only one combo box event: **`OnFinishEditing`** (*see the right-hand-side pane*). If additional events were available 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.



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.



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.

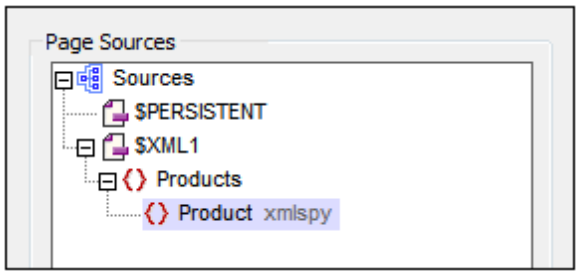
- Warnings indicate design issues that might need your attention, but these issues are not fatal and will not prevent the solution from running.
- An error message, on the other hand—as opposed to a warning—would be fatal and should be remedied.

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.

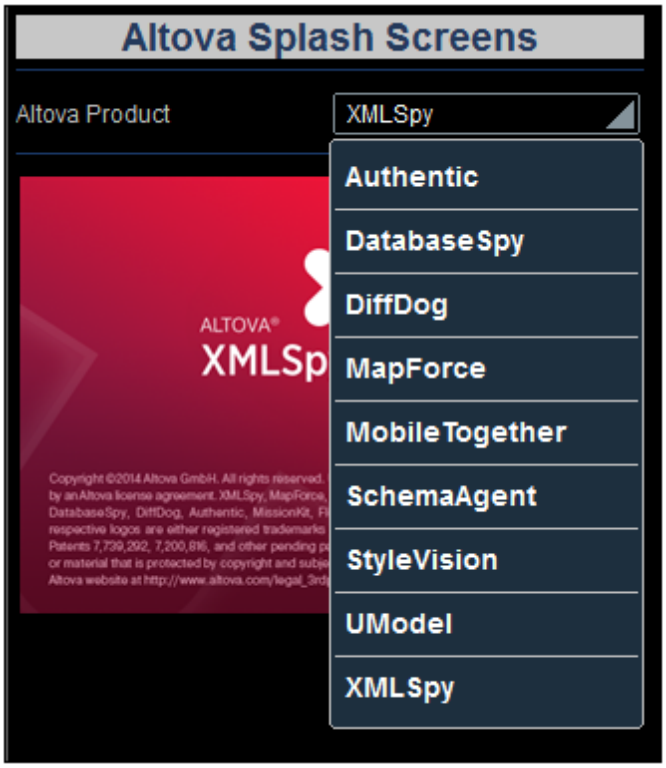
Run a Simulation

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 of the Main toolbar. You can change the preview device to see the simulation on different devices. To run the simulation, select **Project | Simulate Workflow** or **F5**. This opens the simulator and starts the simulation.

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*).



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

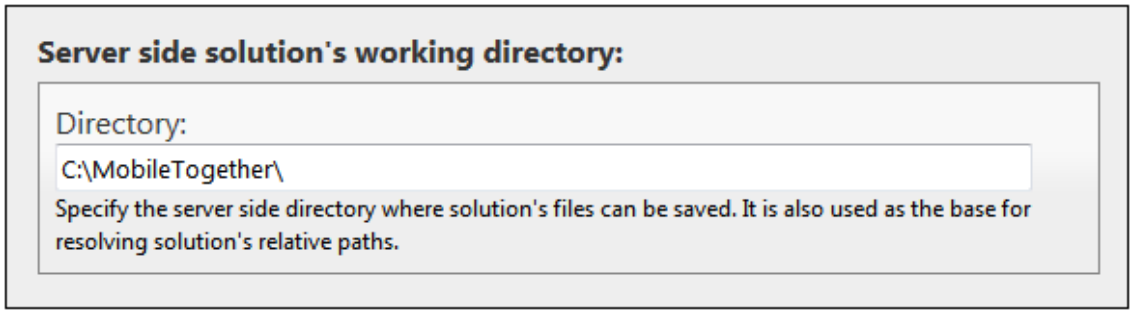


Select a product from the list and notice how the image and value of the **`Product`** node changes. (Note that, for iOS devices, you will need to press **Set** to confirm selections.)

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 (**Project | 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).



Deploy to Server

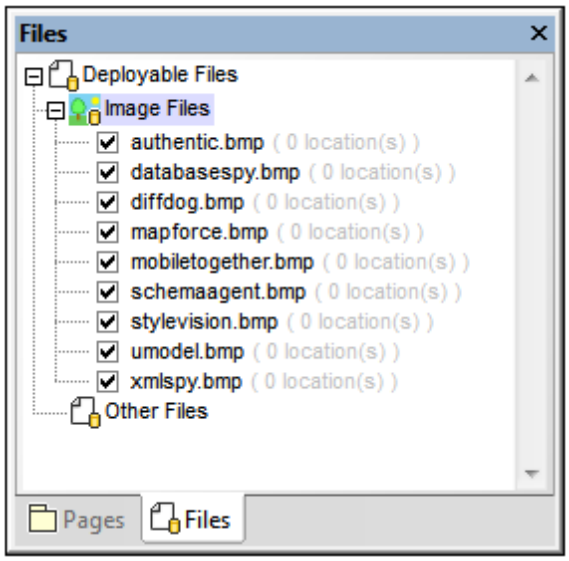
After you have successfully validated the project and tested the simulations, you are ready to deploy the project to MobileTogether Server. When the project is deployed to the server, it becomes a solution that a MobileTogether Client app can run. For more information about deploying the project and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

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.

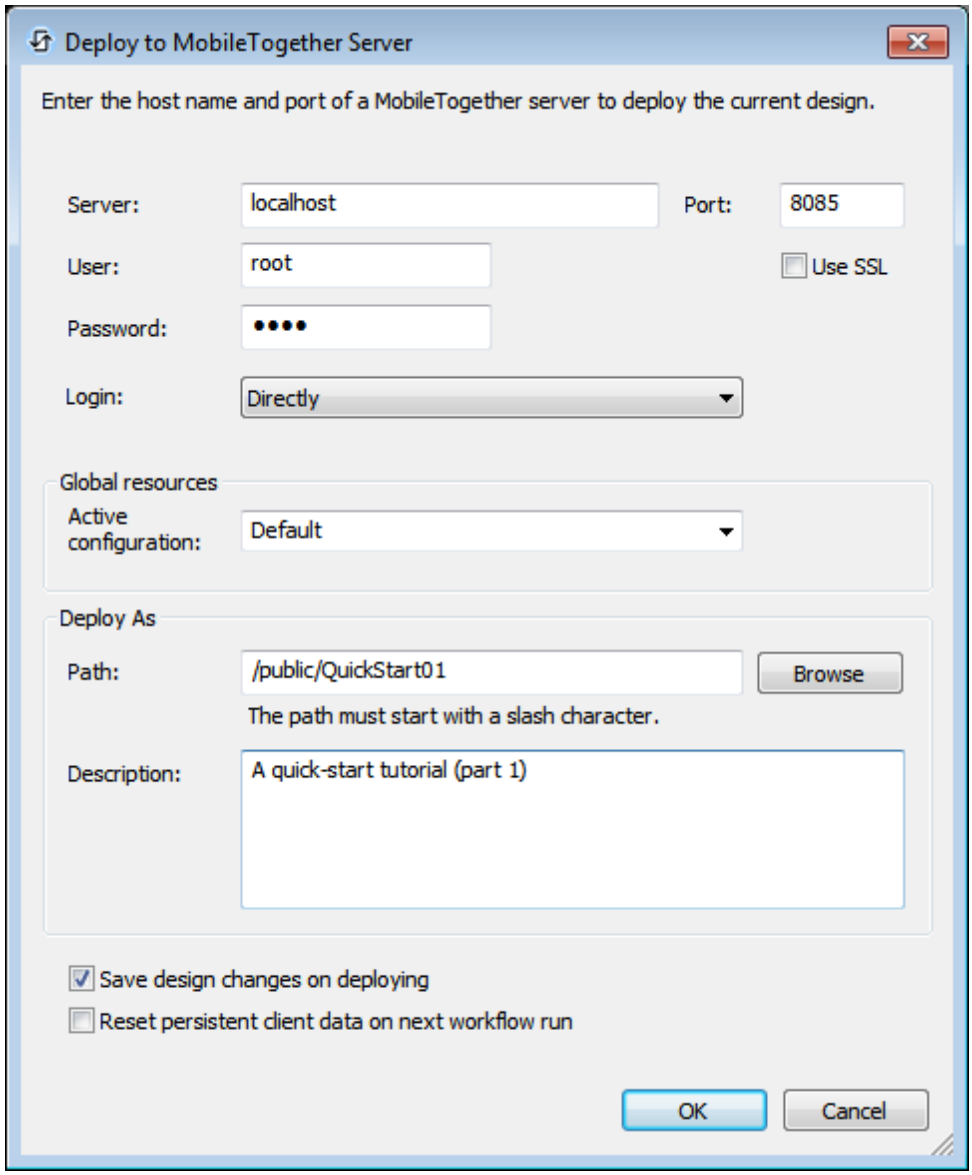
Deploying the project to MobileTogether Server

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

1. In the Files Pane (*screenshot below*), right-click Image Files, and, in the context menu that appears, select **Add File**.



2. Browse for the image file and add it to the Image Files list.
3. Right-click the file and select **Make Relative Path**.
4. Repeat Steps 1 to 3 for all the image files that are required.
5. Select **File | Deploy to MobileTogether Server**. This displays the Deploy to MobileTogether dialog (*screenshot below*).



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 data 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
- Change Source Node
- Run a Simulation
- Use File Data for Combo Box Entries
- Set Data File as Default File
- Create Dynamic Links to Web Pages
- Save Data Back to File

The tutorial files

The files for this tutorial are located in your *(My) Documents* 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`**

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, is called **AltovaProducts.xml** and its listing is given below. It has a similar structure to the data source created in Part 1, with the only difference being that there is one new element: **Selection**.
- Modify the tree structure of the data source to match the different structure of the XML data file.

Listing of the XML file, AltovaProducts.xml

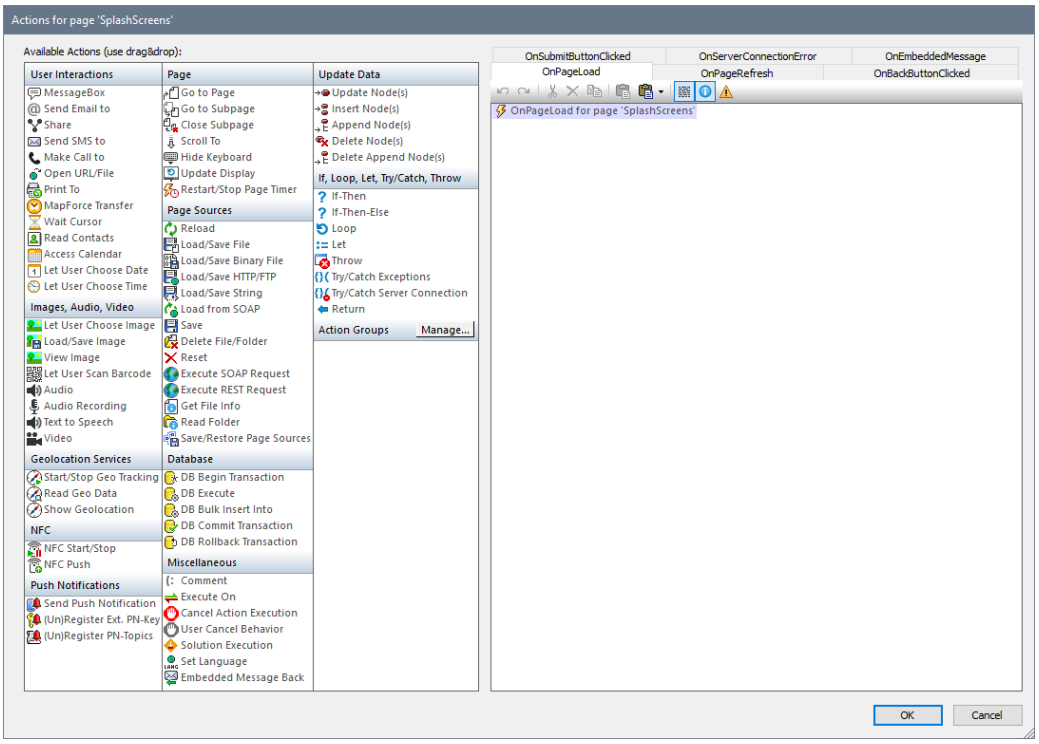
Located in the MobileTogether folder of the *(My) Documents folder*: **MobileTogetherDesignerExamples\Tutorials\AltovaProducts.xml**.

```
<?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>
```

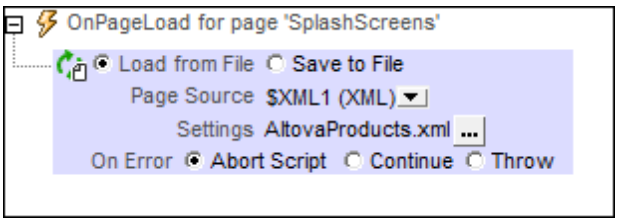
Specify data file to use on page load

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

1. Open **QuickStart01.mtd**, which is located in the MobileTogether folder of the *(My) Documents folder*: **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 **`$XML1`**.
5. Click the **Additional Dialog** button of the **`Settings`** entry. This displays the Specify File dialog.
6. Select *Make path relative to design file* and browse for the file, **`AltovaProducts.xml`**.
7. You will be asked whether you want to deploy this file together with the design file to MobileTogether Server. Click **Yes**. The file will be set as the data file to load for the data source **`$XML1`** when the page is loaded (*screenshot below*).

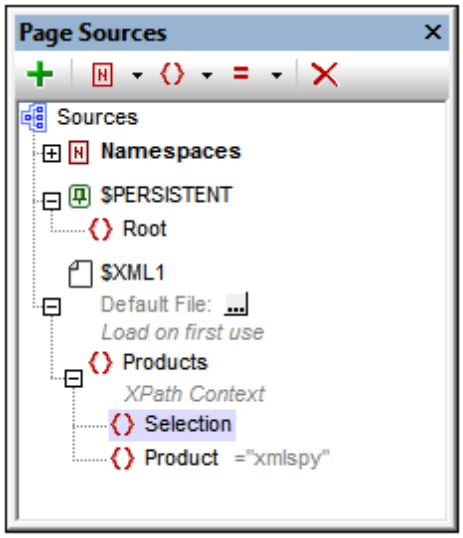


8. Click **OK** to finish.

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 data source in the Page Sources Pane (*see screenshot below and listing above*). 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.

Change Source Node

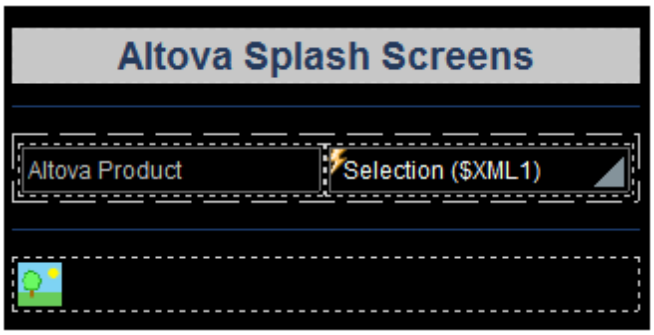
In this part, you will:

- 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 data sources. The link is made by dragging the data source node from the Page Sources Pane 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 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** property (in the Styles & Properties Pane). In the Edit XPath/XQuery Expression Dialog that appears, modify the XPath expression so that **Product** is replaced by **Selection**. For example:

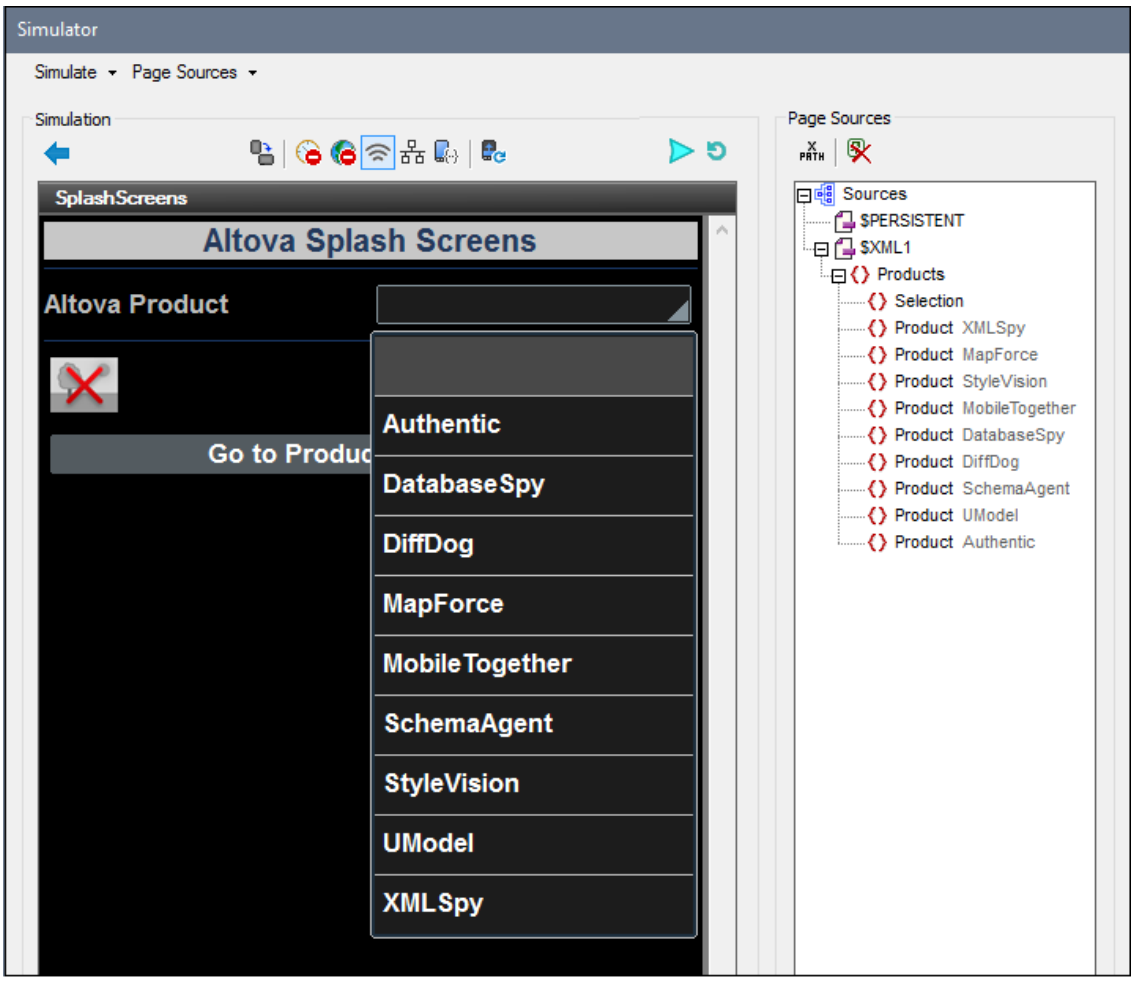
If you have: `concat(Product, '.bmp')`

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.

Run a Simulation

If you run a simulation now (**Project | 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
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 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*).

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.

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.

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

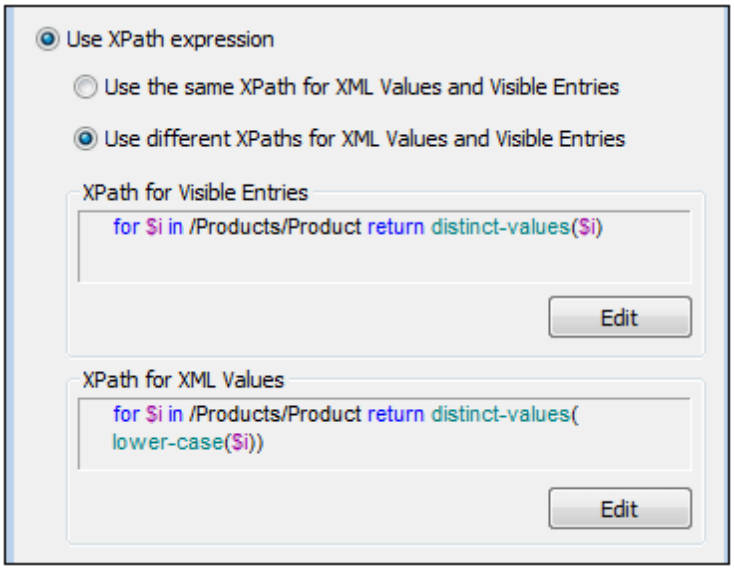
Located in MobileTogether folder of *(My) Documents folder*:
MobileTogetherDesignerExamples\Tutorials\AltovaProducts.xml.

```
<?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, click the Additional Dialog button of the **Combo Box Entry Values** property. The Edit Combo Box dialog (*screenshot below*) appears.



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. 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). 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.

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.

Set Data File as Default File

In this part, you will:

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

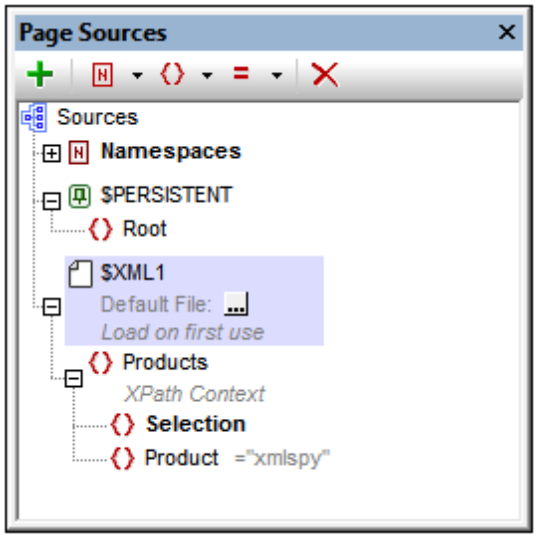
Buttons in this section

... **Additional Dialog**

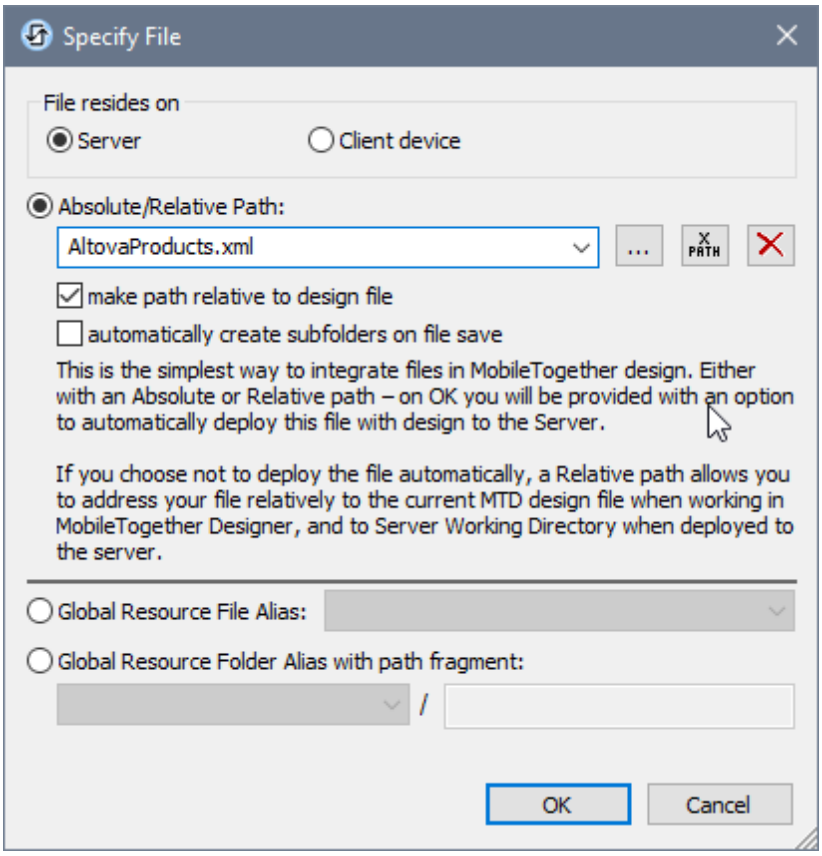
Specify a default file for the page data source

The data that goes into a data source can be specified by selecting a default file for the data 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**.



3. On clicking **OK**, the file will be added as the default file, and its data will populate the data source tree.
4. Click **Page | Page Actions** to open the Page Actions dialog.
5. In the `OnPageLoad` tab, select the Load from File entry 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 data source.
6. Run a simulation to test whether the default file is being used.

Create Dynamic Links to Web Pages

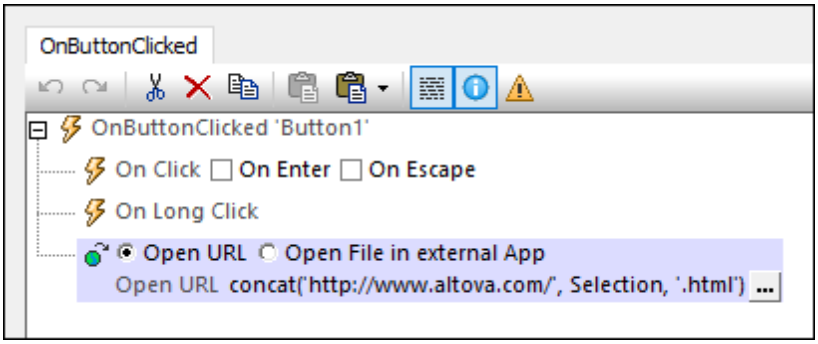
In this part, you will:

- 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 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.
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')`



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

Run a simulation

Run a simulation by clicking **F5** or **Project | 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.


SplashScreens
Altova Splash Screens
Altova Product
StyleVision
ALTOVA®
StyleVision® 2014
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.
Go to Product Description

Save Data Back to File

In this part, you will:

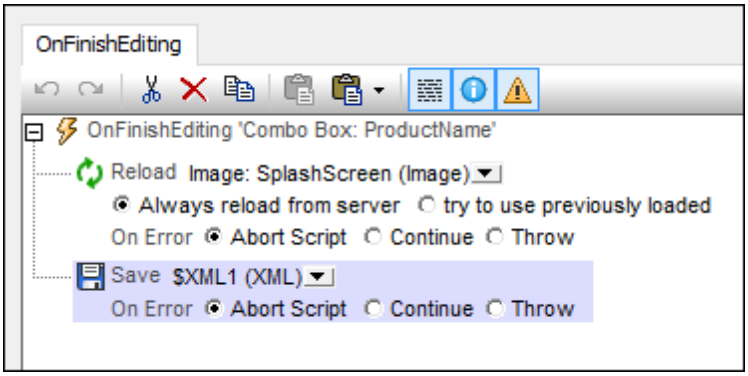
- Save changed data back to file using a control action
- Run a simulation to test the success of the *Save to File* action

---

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, the combo box selection will be saved to this element. To specify that the 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 below the *Reload* action, and drop (*see screenshot below*).



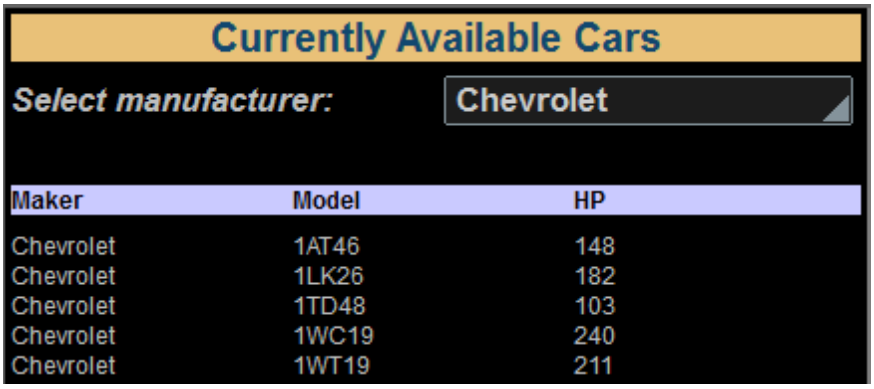
4. Click **OK** to finish, and then save the file.
5. To test whether the change is saved to the default file, open the default file in an editor, run a simulation, select a combo box entry, and then reload the default file in the editor. The new combo box selection will appear as content of the `Selection`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



The tutorial files

The files for this tutorial are located in your *(My) Documents* 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
- Persistent Tree for User Input
- Load DB Data Based on User Selection

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.

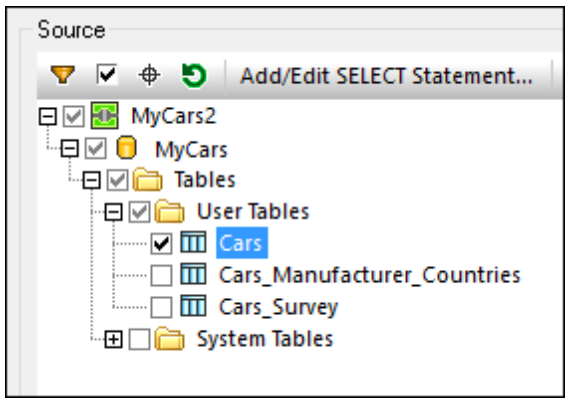
The structure of the DB is as shown in the screenshot below. (Note that the screenshot shows only the first few records of the DB.)

	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:

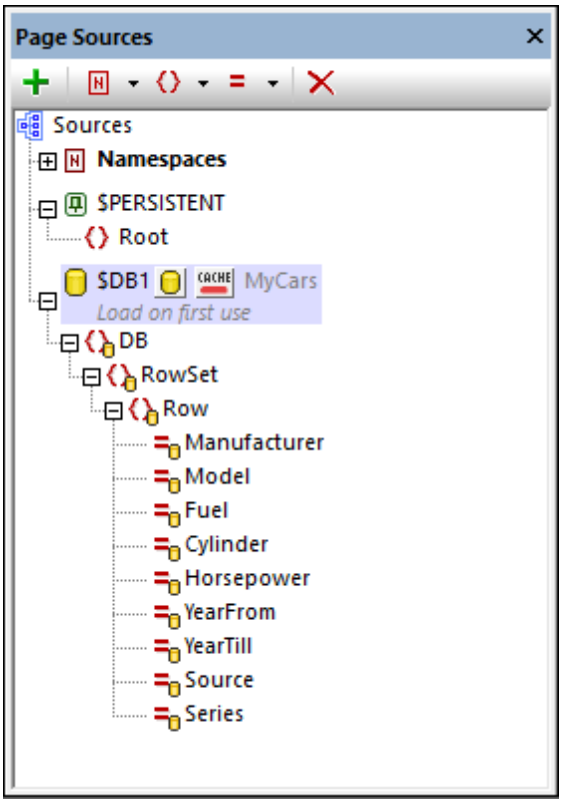
1. Create a new design (**File | New**).
2. Create a data source by clicking the **Add Source** icon in the Page Sources pane.
3. In the Add Page Source dialog that appears, select *New DB Structure* and then **Next**.
4. In the next screen of the dialog, leave the default settings as they are and click **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 the Insert Database Objects dialog (*screenshot below*) appears.



Select the `Cars` table as shown in the screenshot, and click **OK**.

The `Cars` table is created as a DB data source of the design page, and is displayed in the Page

Sources pane (*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.

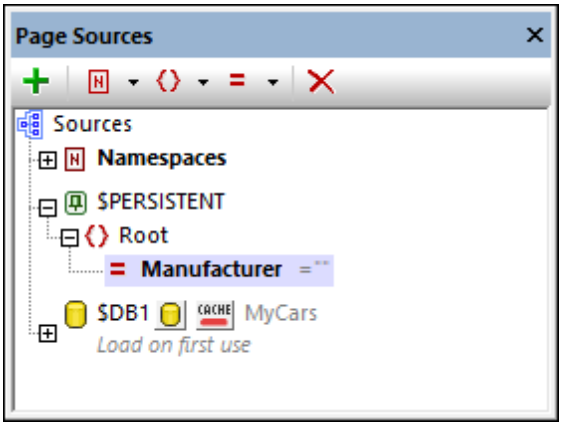
Persistent Tree to Hold User Input

To interact with the end user, a combo box 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 to display the selected manufacturer's cars (*see next section*).

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 **$PERSISTENT** tree is automatically created in the Page Sources pane with a root (or document) element called **Root** (*see screenshot below*). Since only a single node is required to hold the user selection, all that is needed is a child element or child attribute node. Add an attribute node called **Manufacturer** to the **Root** element as follows. Select **Root** in the **$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:

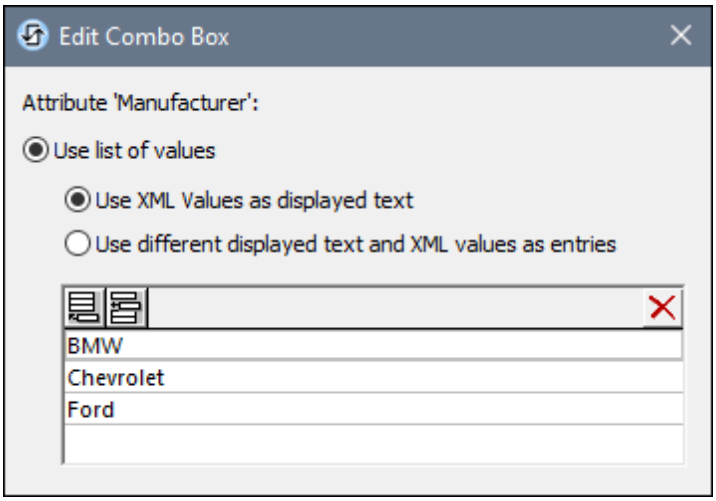
The design at this point should contain:

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

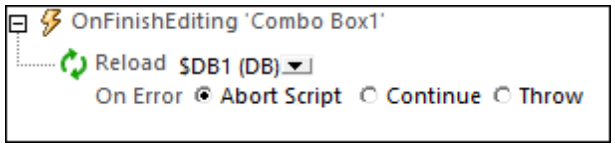
To add the label, table, and combo box controls, drag the respective controls from the Controls Pane. For details about each control's properties, see the description of the respective control.

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.



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



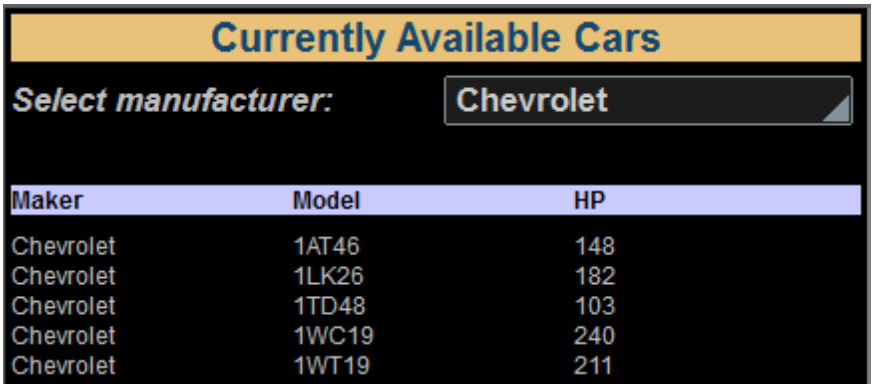
Next, we will load the selected manufacturer's cars from the DB into the solution, and display these cars in a table.

Load DB Data Based on User Selection

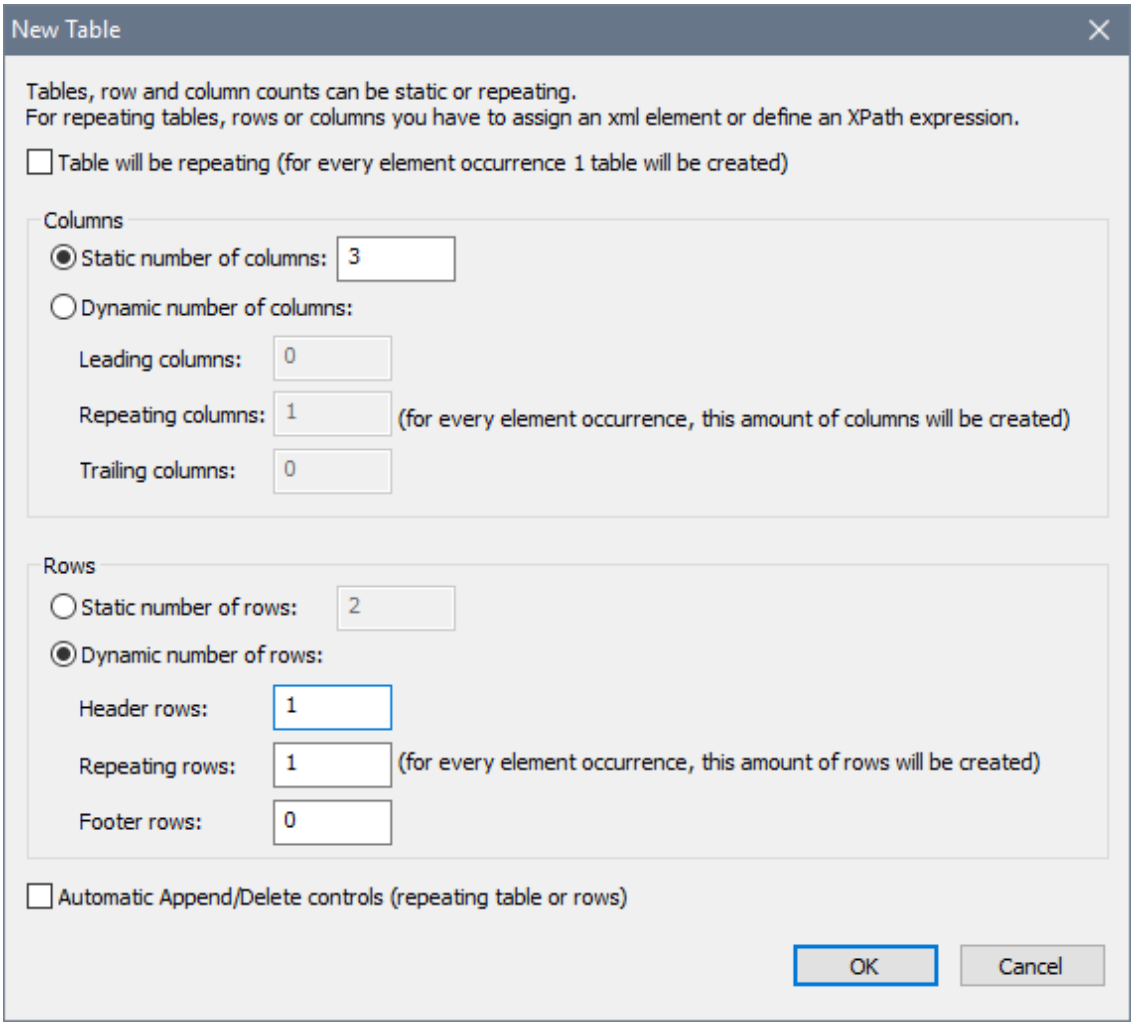
The DB records of car models will be displayed in a table, with only those models being loaded and displayed that belong to the manufacturer chosen by the end user (in the combo box). See the previous section for details of how this is set up. In this section, first, a table is created to display the 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.



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.



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 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, and create them as label controls: **Manufacturer**, **Model**, **Horsepower** (*see screenshot below*).

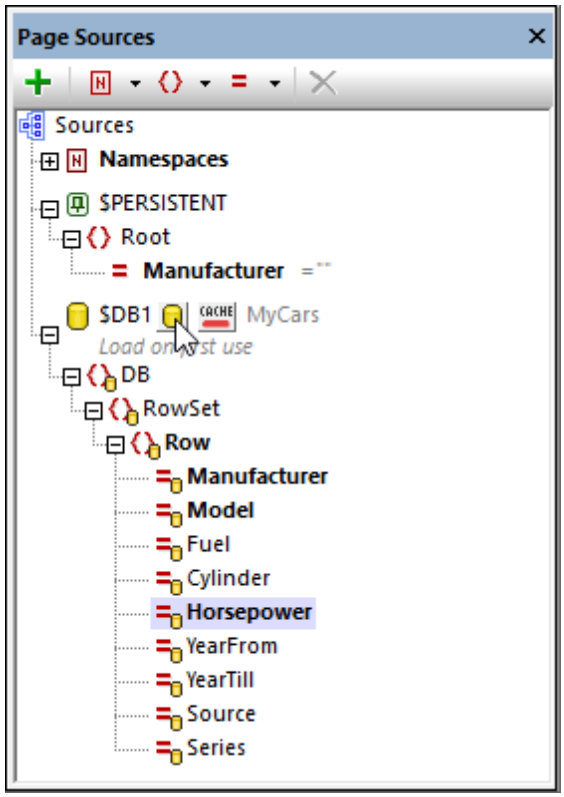
When you have finished, the table in the design should look something like this:

Select the DB records to load and display

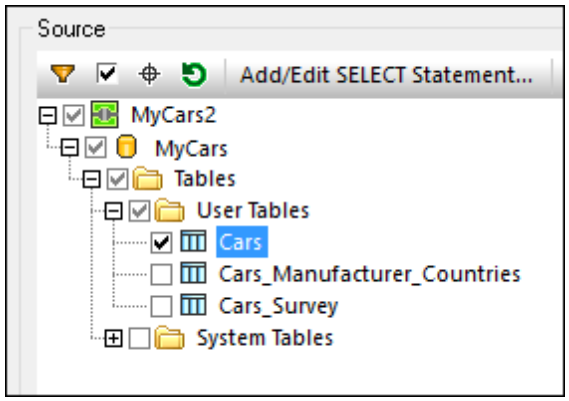
The table that you have just created will display all the records that are loaded from the DB. And the way the **$DB1** tree is currently defined, **all the records in the DB** will be loaded—that is, all the car models of **all the manufacturers**—and all will be displayed. However, the goal is to **load**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

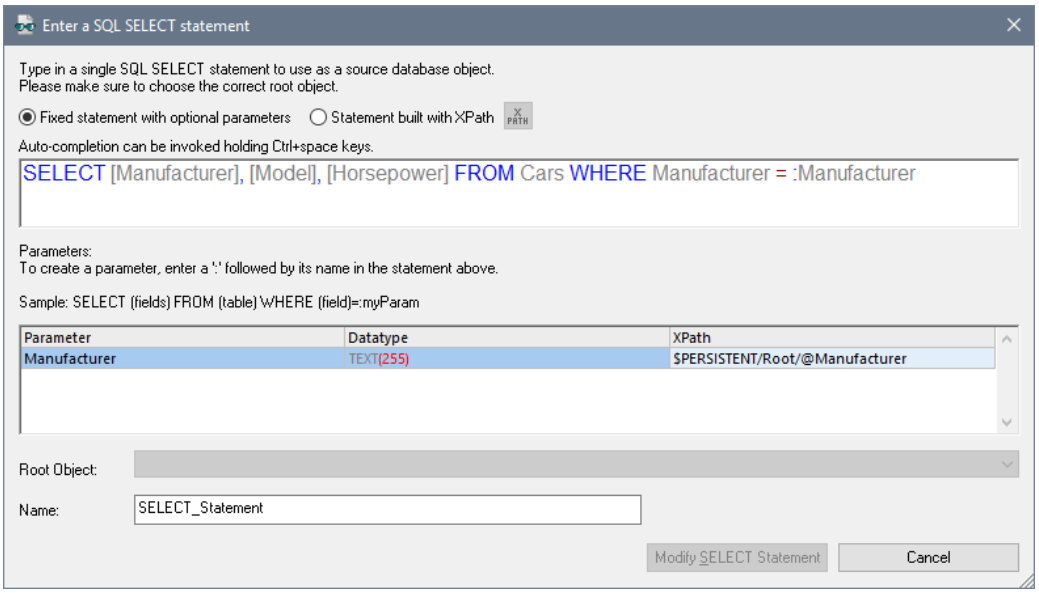
1. Click the DB icon to the right of the **$DB1** tree legend (*see cursor in the screenshot below*).



2. In the Insert Database Objects dialog that appears (*screenshot below*), click **Add/Edit SELECT Statement**.



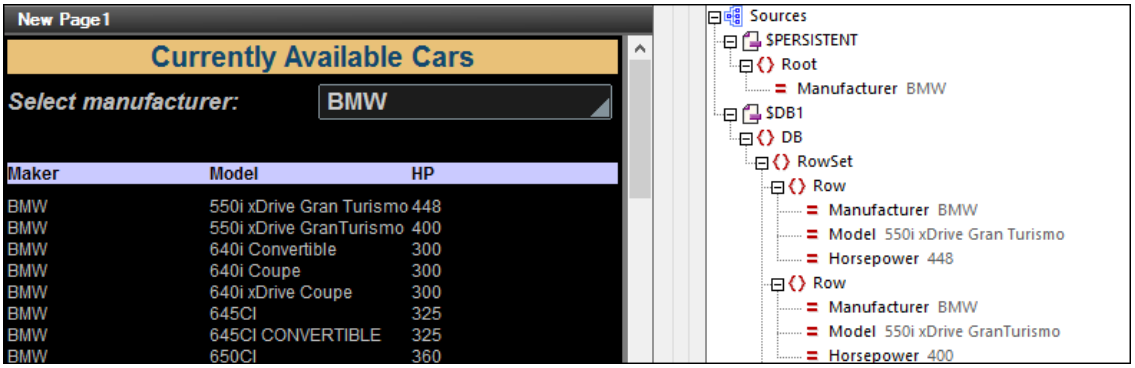
3. In the Enter an SQL SELECT Statement dialog that appears, enter the following **SELECT** statement: `SELECT [Manufacturer], [Model], [Horsepower] FROM Cars WHERE Manufacturer = :Manufacturer`. This statement selects—and therefore loads—only the **Manufacturer**, **Model**, and **Horsepower** fields 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*).



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 `$DB1` data source selects what data from the DB to **load** into the `$DB1` tree. This is how the mechanism works:

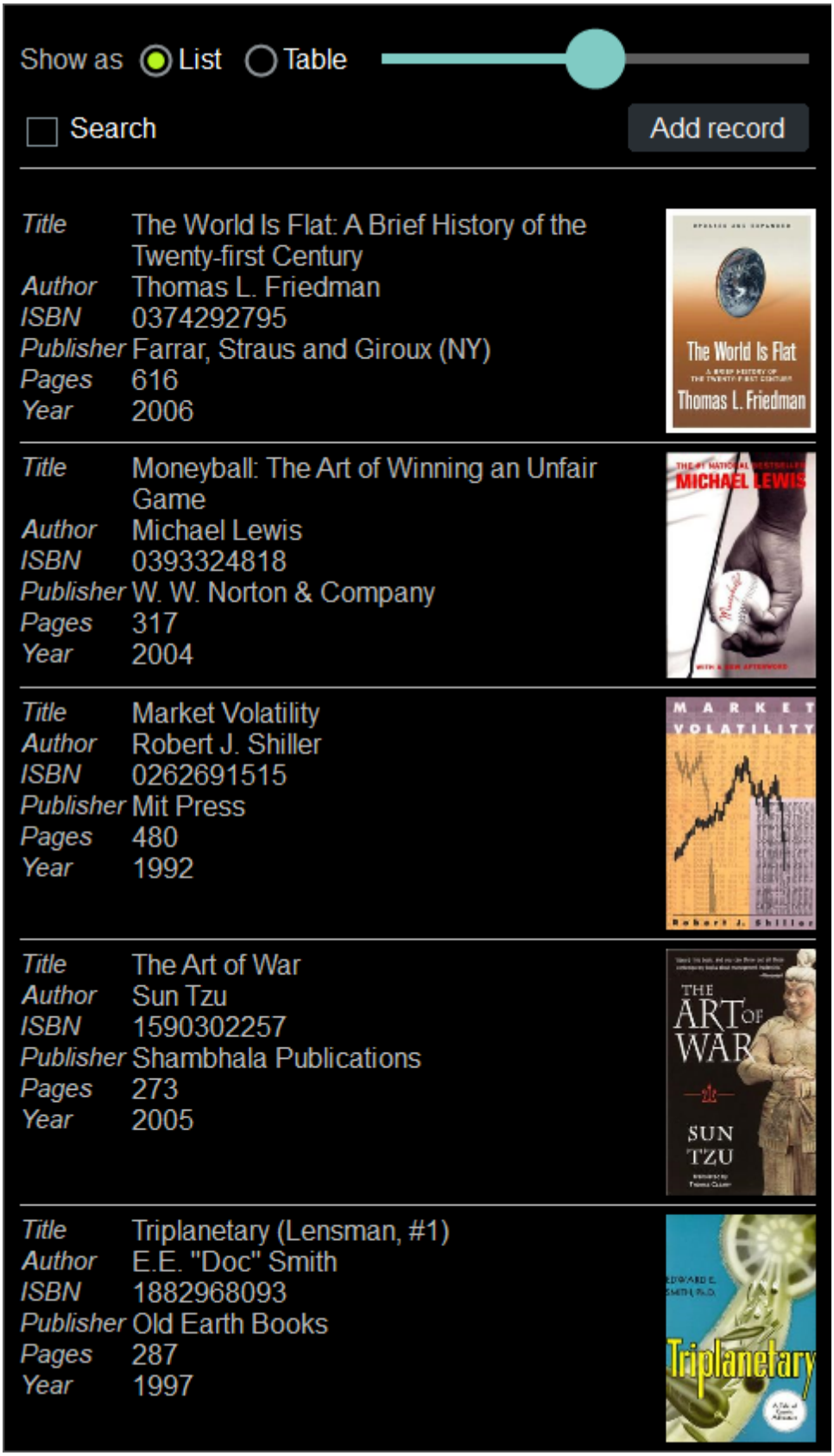
- As soon as the user changes the value in the combo box, the Reload action of the control's **OnFinishEditing** event reloads the `$DB1` tree (see the definition of the combo box).
- The `$DB1` tree is loaded on the basis of a **SELECT** statement.
- This **SELECT** statement uses a parameter that has 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.


New Page1
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 Advanced Database

This tutorial describes key features of a DB-based design (*screenshot below*). The SQLite DB is a catalog of books. The design does the following:

- displays the (book) records, together with images of the book covers
- enables the records to be edited (including the modification of cover images in the DB)
- enables new records (including cover images) to be added
- enables the DB to be searched and the search results to be displayed


Show as List Table
Search
Add record
Title The World Is Flat: A Brief History of the Twenty-first Century
Author Thomas L. Friedman
ISBN 0374292795
Publisher Farrar, Straus and Giroux (NY)
Pages 616
Year 2006
The World Is Flat
Thomas L. Friedman
Title Moneyball: The Art of Winning an Unfair Game
Author Michael Lewis
ISBN 0393324818
Publisher W. W. Norton & Company
Pages 317
Year 2004
MICHAEL LEWIS
Title Market Volatility
Author Robert J. Shiller
ISBN 0262691515
Publisher Mit Press
Pages 480
Year 1992
MARKET VOLATILITY
Robert J. Shiller
Title The Art of War
Author Sun Tzu
ISBN 1590302257
Publisher Shambhala Publications
Pages 273
Year 2005
THE ART OF WAR
SUN TZU
Title Triplanetary (Lensman, #1)
Author E.E. "Doc" Smith
ISBN 1882968093
Publisher Old Earth Books
Pages 287
Year 1997
Triplanetary

The tutorial files

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

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

You can copy these files to any folder, open the design file in MobileTogether Designer, and run simulations in MobileTogether Designer.

Tutorial structure

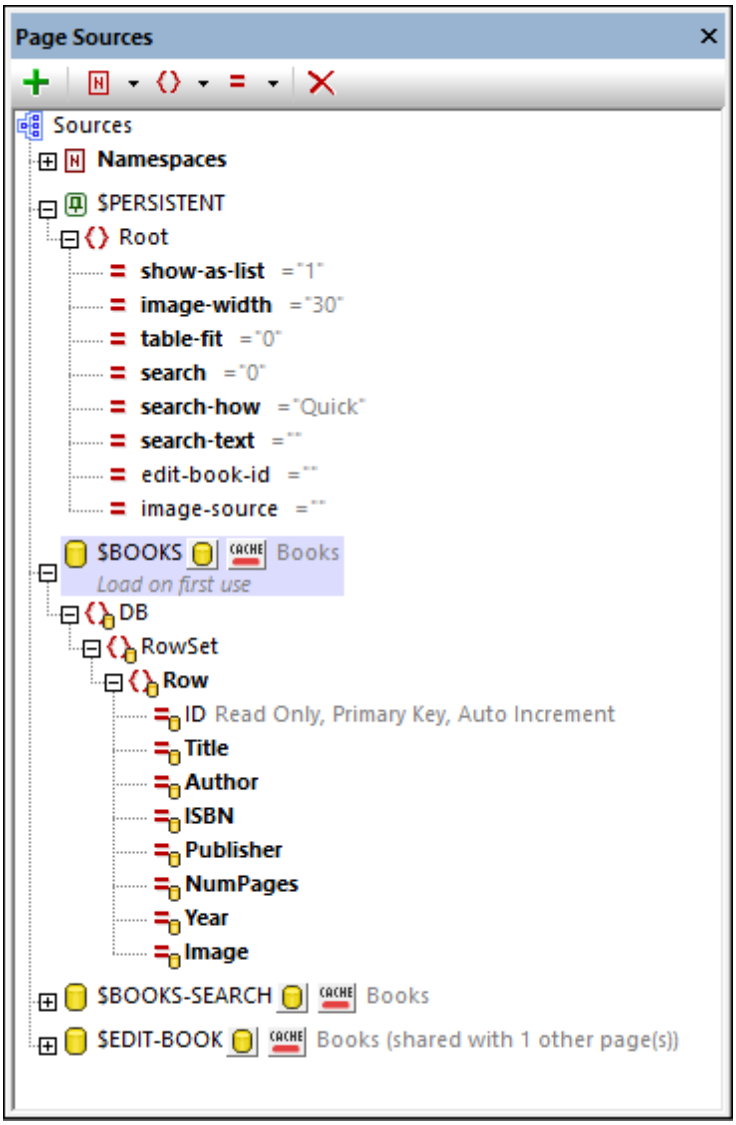
This tutorial is organized into the following sections:

- Page Sources: DB and Persistent
- Controlling Visibility
- Images in the DB
- Editing Records
- Adding Records

Page Sources: DB and Persistent

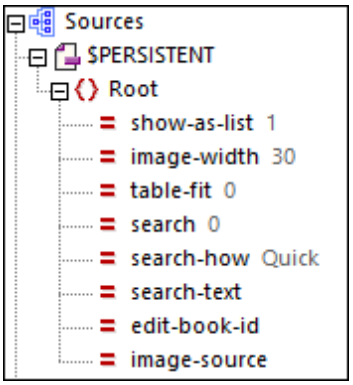
The design has two pages: (i) a top page named *Main Page*, and (ii) a sub page named *Edit Book*.

- *Main Page* displays the books, either (i) unfiltered, or (ii) filtered according to a search term. On *Main Page,* the end user can also: (i) click a record in order to edit it, and (ii) click a button to add a new record. The click in both cases accesses the *Edit Book* sub page. *Main Page* has four page sources (*see screenshot below*): **$PERSISTENT**, **$BOOKS**, **$BOOKS-SEARCH**, **$EDIT-BOOK**.
- The *Edit Book* sub page contains a template containing the fields of the record. If a record is being edited, the template contains the book details. If a record is being added, the template is empty. *Edit Book* has two page sources: **$PERSISTENT** and **$EDIT-BOOK**. It shares **$EDIT-BOOK** with *Main Page*.



$PERSISTENT

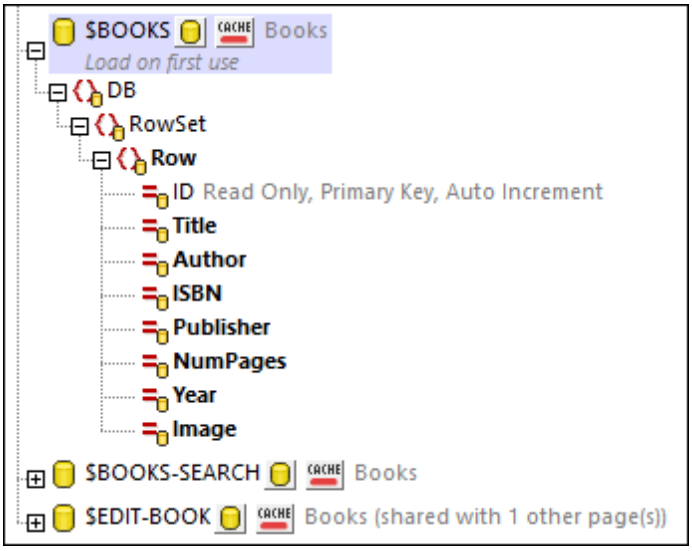
The **$PERSISTENT** page source is used to store dynamic data. It has a single element named **Root** that has a number of attributes (*see screenshot below*), the values of which are determined by user actions. For example, the attribute **show-as-list** takes a value of **1** or **0**, depending on whether the user selects, respectively, the *List* or *Table* radio button.

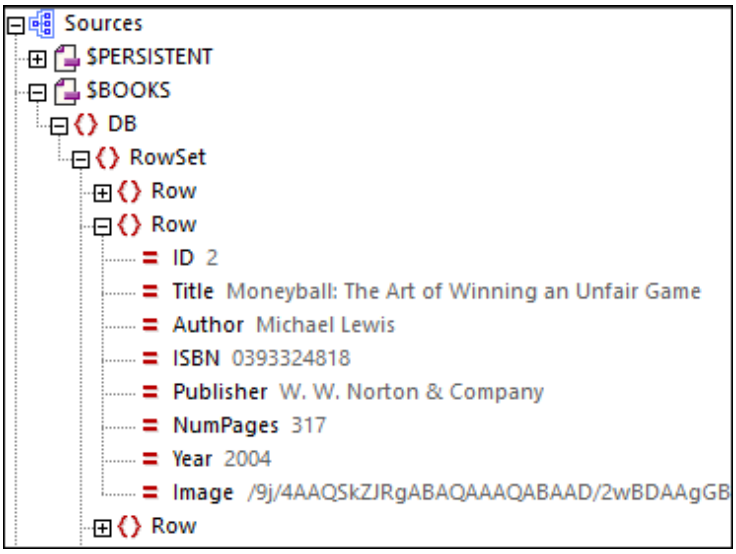


The values in the **$PERSISTENT** page source are used to conditionally determine various aspects of the design. For example, when the value of **show-as-list** is **1**, then the book catalog is displayed as a list, and not as a table.

$BOOKS

The **$BOOKS** page source selects the **Books** table from the SQLite DB **Books.sqlite**. Each book record corresponds to a **Row** element in the page source, with the fields (or columns) of each DB row corresponding to the attributes of the respective **Row** element. This is shown in the screenshots below of the page source in the design (*left*) and in a simulation (*right*).

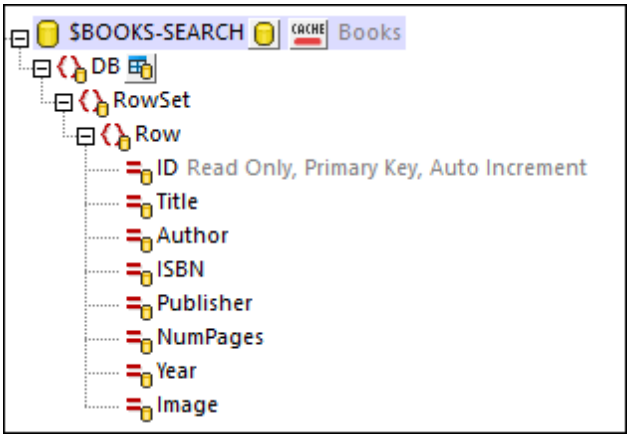


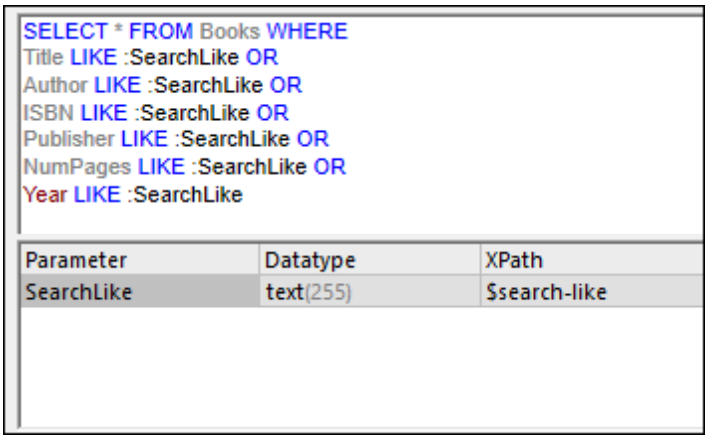


The **$BOOKS** page source is used to display the entire catalog of books.

$BOOKS-SEARCH

The **$BOOKS-SEARCH** page source (*screenshot below left*) filters the **Books** table from the SQLite DB **Books.sqlite** to select books where the value of any field in the record matches the search term at least partially. In the screenshot below left, notice the filter, which is defined in the page source on the root element **DB**. The SQL **SELECT** statement of the filter is shown in the screenshot below right. The **:SearchLike** parameter in the **SELECT** statement resolves to the search term that the end user enters.

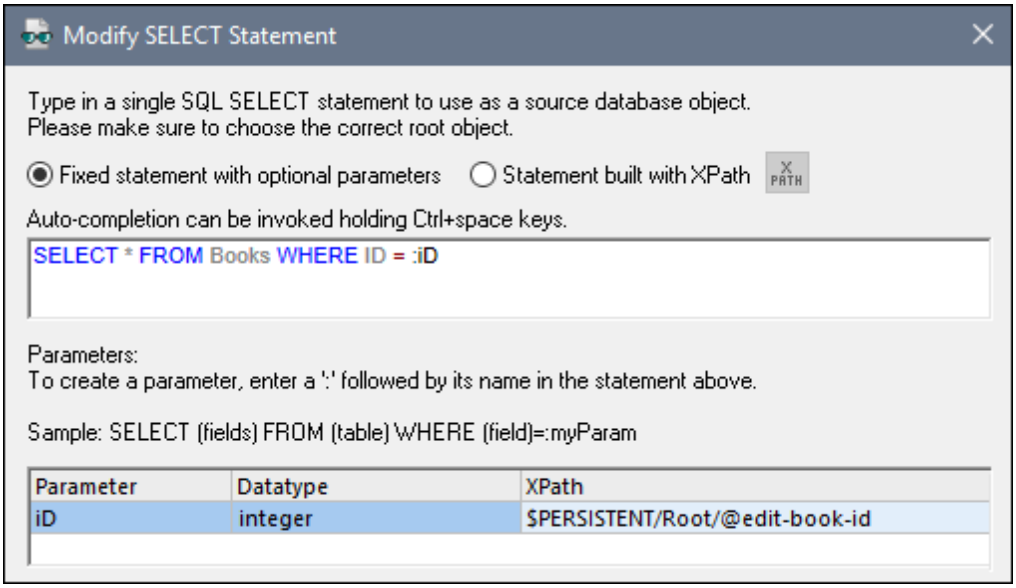




The **$BOOKS-SEARCH** tree replaces the **$BOOKS** tree in the display of the top page during a search.

$EDIT-BOOK

The **$EDIT-BOOK** page source is similar to the **$BOOKS-SEARCH** page source in that it filters the **Books** table from **Books.sqlite**. The SQL **SELECT** statement of the filter is shown in the screenshot below. The **:iD** parameter in the **SELECT** statement resolves to the value in the node **$PERSISTENT/Root/@edit-book-id**, which is the ID of the book record being edited or added.



The **$EDIT-BOOK** page source is used as the data link of the editing template in the *Edit Book* sub page. The **$EDIT-BOOK** page source also contains an **OriginalRowSet** node (which must be created, via the context menu) to hold the original data while **RowSet** holds the current (edited) data (*see screenshot below*).

Controlling Visibility

The design of *Main Page* uses the Visible property of tables and table columns to control the layout by switching on/off the display of table components depending on end user actions.

User-driven display of one table in a set

The design of *Main Page* consists of three tables, which are, from top to bottom:

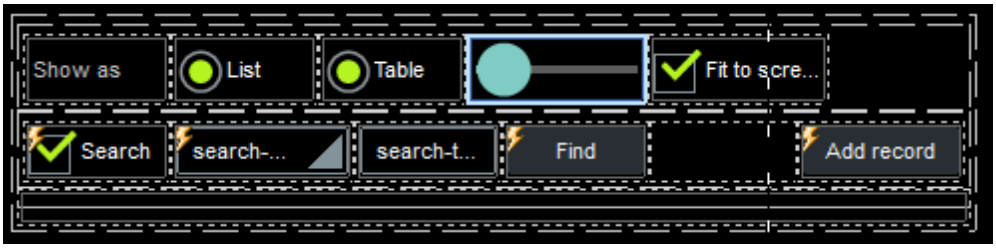
- To hold controls for user interaction that will determine the way data is displayed
- To display the book catalog as a list, visible when the end user selects the *List* radio button
- To display the book catalog as a table, visible when the end user selects the *Table* radio button

The book catalog is displayed as a list or as a table, not both. This is defined using he following mechanisms:

1. When the radio button choice is made (*List* or *Table*), the node **`$PERSISTENT/Root/@show-as-list`** receives a value of **`1`** for *List* or **`0`** for *Table*.
2. The **`Visible`** property of the table (in the design) that displays the book catalog as a list is set to: `$PERSISTENT/Root/@show-as-list = 1`. So this table will become visible only when the radio button choice is *List*.
3. The **`Visible`** property of the table that displays the book catalog as a table is set to: `$PERSISTENT/Root/@show-as-list = 0`. So the visibility of this table is switched on only when the radio button choice is *Table*.

Conditional display of table columns

The topmost table of *Main Page (screenshot below)* consists of two one-row tables. The display of several columns in these two table have been made conditional on certain user selection. To see these settings. click inside a column and then look at the XPath expression of the **`Visible`** property of that table column.



The visibility conditions of some of these columns are given below:

- The column of the Horizontal Slider (*highlighted in blue above*) is visible when the *List* radio button is selected, invisible when the *Table* radio button is selected.
- The column of the *Fit to Screen* check box is visible when the *Table* radio button is selected, invisible when the *List* radio button is selected.

- The columns of the *Search How* combo box and *Search Term* edit field are visible when the *Search* check box is selected, invisible when it is not selected.
- The column of the *Add Record* button is visible when the *Search* check box is not selected, invisible when it is selected.

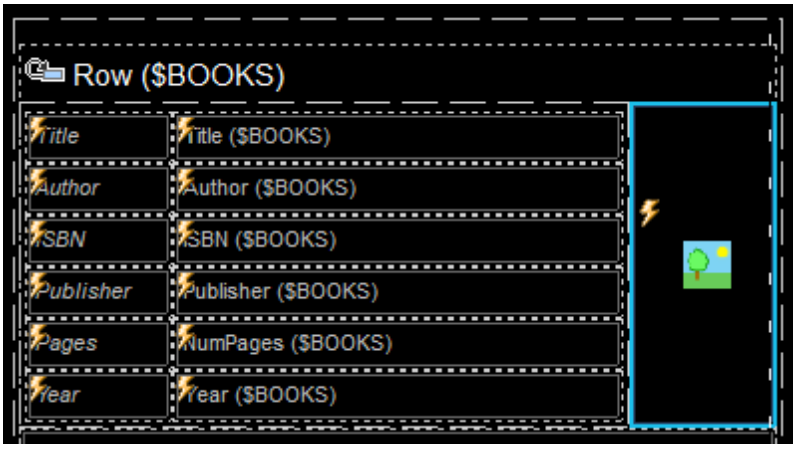
Images in the DB

An important point to note at the outset is that images are stored **in DBs and page source nodes** as **Base64-encoded text strings**.

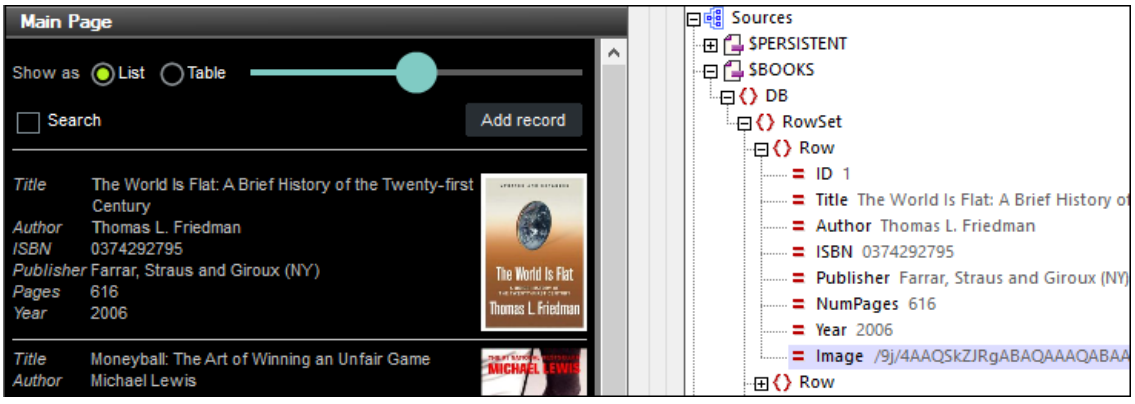
Note: An image can be loaded directly into the client display from an image file (such as PNG or JPEG) if the file does not pass through a page source node on the server. For more information, see (i) the description of the Image control and (ii) the section Images.

Displaying images from a DB

Since images are stored in databases as a Base64-encoded string, the display is straightforward: the Image control in the list display of the book catalog (*see screenshot below*) specifies (i) **base64** as the value of the control's `Image Source Type` property, and (ii) the node containing the Base64-encoded string.

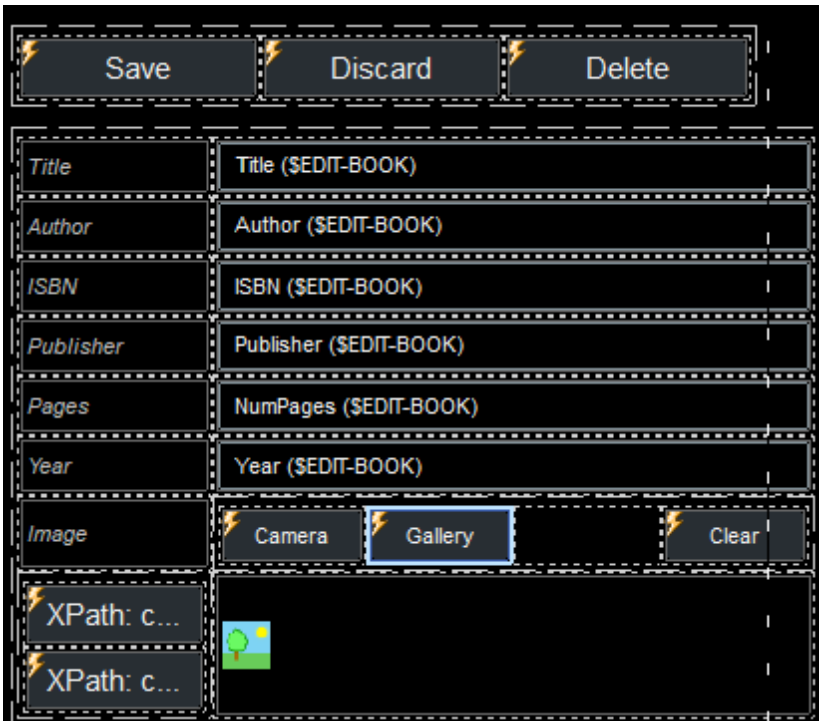


In a simulation (*shown below*), the Base64-encoded string (*highlighted*) can be seen in the page source tree.

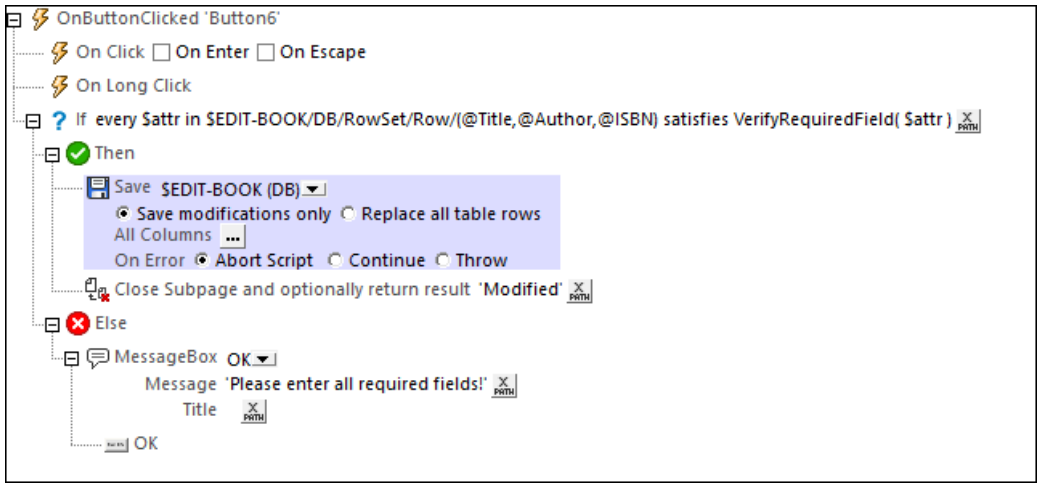


Saving images to a DB

To save an image to the DB, the following steps are taken. Note that the image must be saved to the DB as a Base64-encoded string. You can follow these steps by looking up the actions on the *Edit Book* sub page.

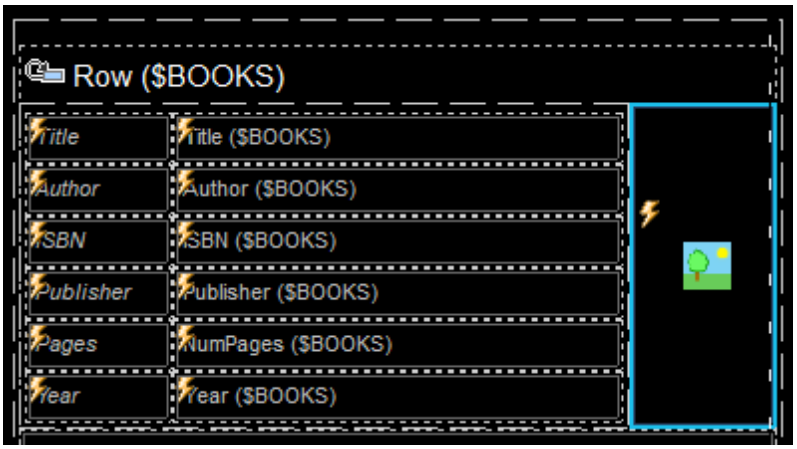


1. An image is selected from the device's gallery or is the next photograph taken by the device's camera. The Let User Choose Image action is used for this step. See the actions of the **Gallery** and **Camera** buttons (*refer screenshot above*).
2. The two buttons to the left of the Image control use the **`mt-transform-image`** function to rotate the image counter-clockwise and clockwise by 90 degrees, respectively.
3. The image is saved to a node of the **`$EDIT-BOOK`** page source as a Base64-encoded string. The Let User Choose Image action automatically stores the image data as a Base64-encoded string.
4. Once the image data is in a page source node, the node can be saved to the DB. In our design, this is done by saving the page source (**`$EDIT-BOOK`**) to the DB. If the record is a new record, it is added to the DB. If the record is an existing record that has been edited, then it replaces the existing record. See the actions of the **Save** button (*shown in screenshot below*).

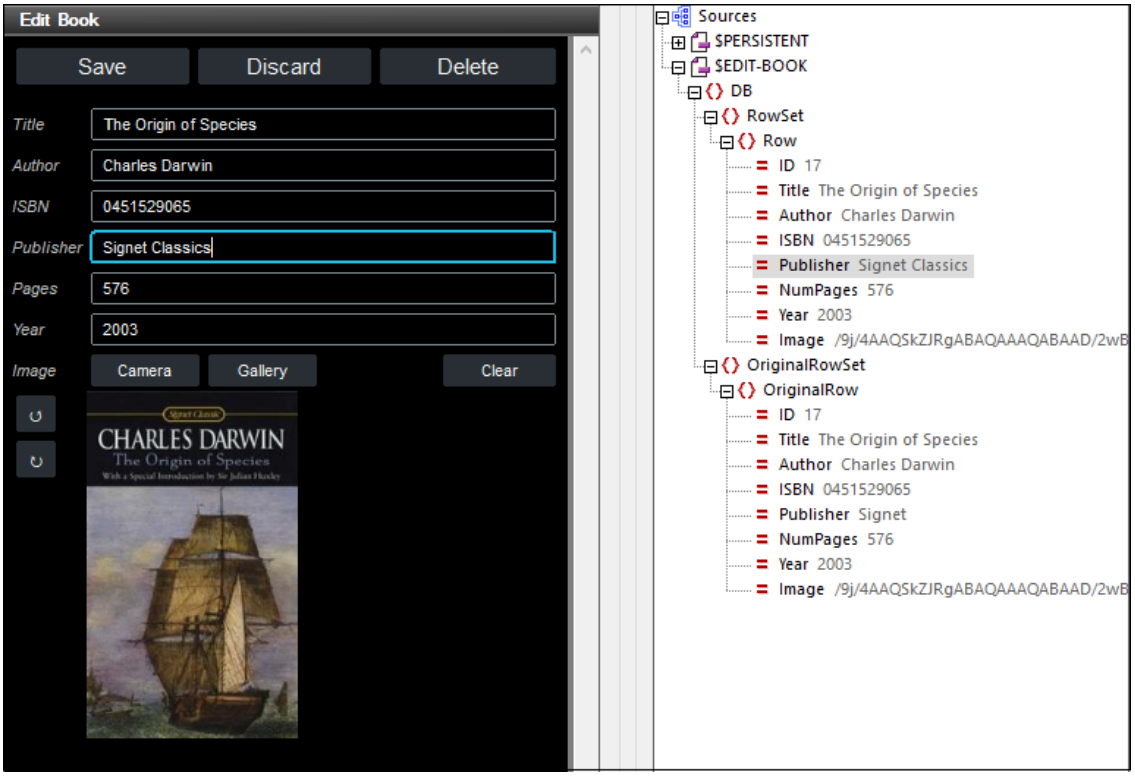


Editing Records

To edit a record displayed on the top page, the record is selected by clicking one of its display fields. Each field has the same set of actions defined for it (*see screenshot below*), which is essentially to go to the *Edit Book* sub page. You can follow this by looking up the actions of one or more of these fields.



The selected book will be displayed in the editing template of the *Edit Book* sub page (*shown below in a simulator*).

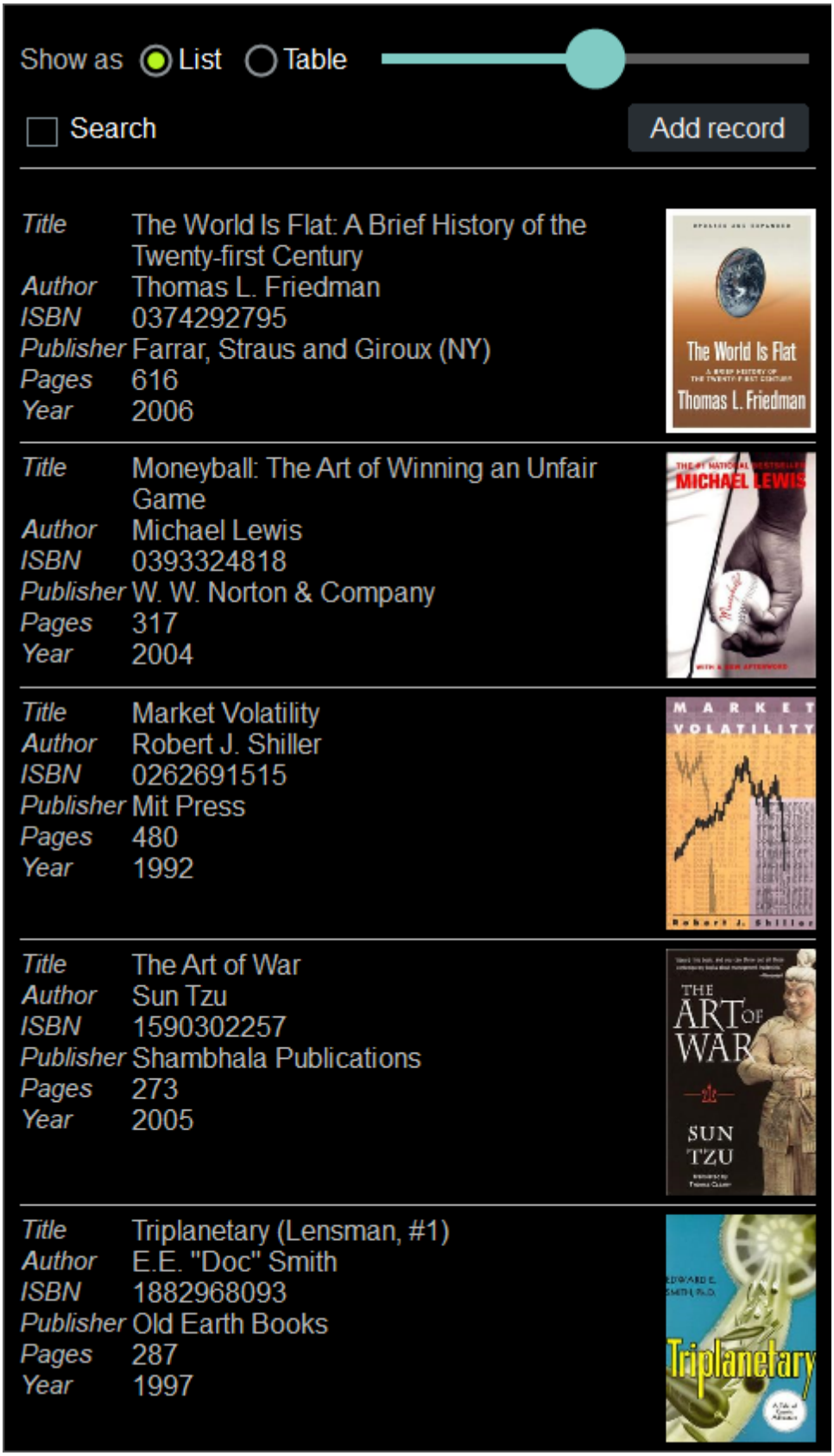


Note the following points:

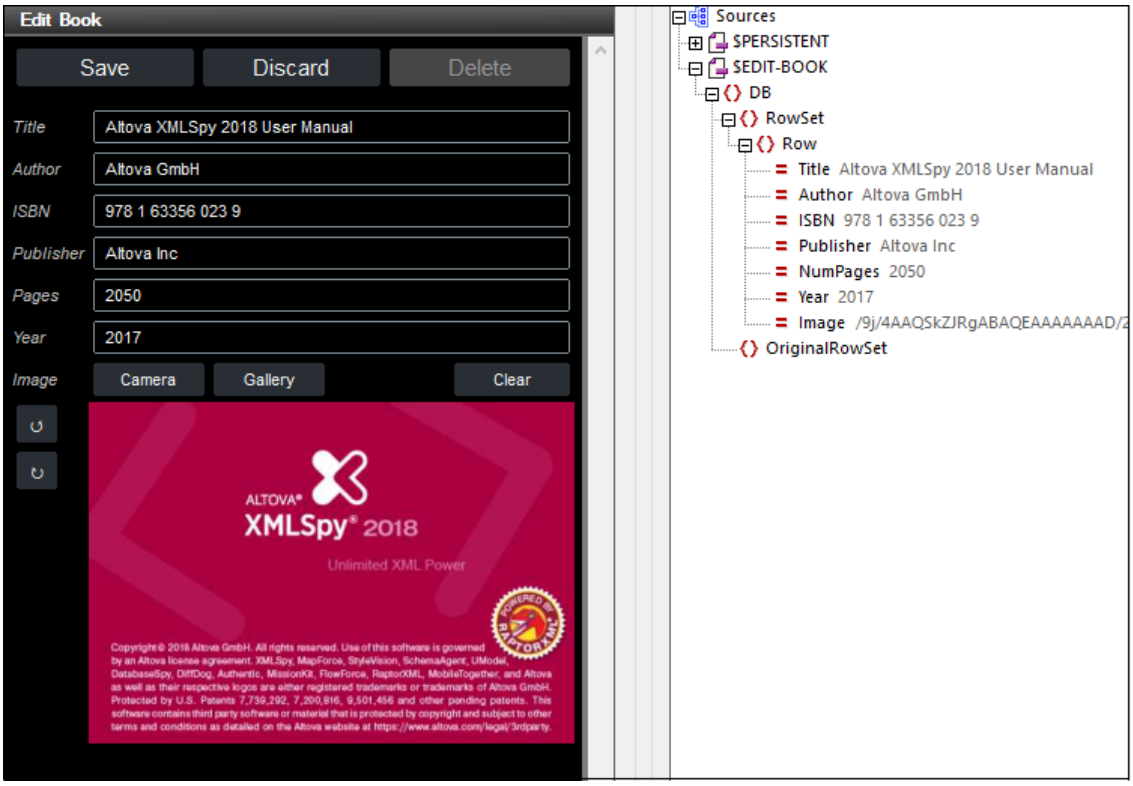
- The page source of the *Edit Book* sub page is **`$EDIT-BOOK`**, which filters the records of the DB to show only the selected book.
- The page source contains an **`OriginalRowSet`** node to hold the original data while **`RowSet`** holds the data being edited. In the screenshot above, the *Publisher* field has been edited from *"Signet"* to *"Signet Classics"*. Compare the Publisher fields of **`OriginalRowSet`** and **`RowSet`**.
- If the **Save** button is clicked, the data from **`RowSet`** is saved to the DB. See the actions of the **Save** button.
- If the **Discard** button is clicked, the data from **`RowSet`** is discarded. See the actions of the **Discard** button.
- How images are handled is described in the section Images in the DB.

Adding Records

To add a record, click the **Add Record** button of the top page (*see screenshot below*).



The action of the **Add Record** button takes you to the *Edit Book* sub page (*screenshot below*), where the details of a new book record can be added and an image selected for storage in the DB.



Note the following points:

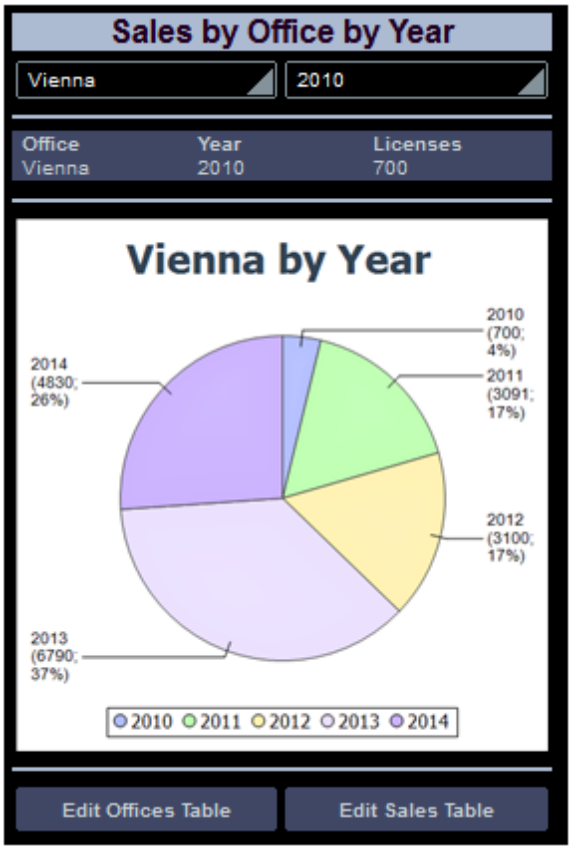
- The page source of the *Edit Book* sub page is **`$EDIT-BOOK`**, which filters the records of the DB to show only the selected book.
- The page source contains an **`OriginalRowSet`** node because the node has been defined for the page source, but the node is empty. This node is used when an existing record is being edited.
- If the **Save** button is clicked, the data from **`RowSet`** is saved to the DB. See the actions of the **Save** button.
- If the **Discard** button is clicked, the data from **`RowSet`** is discarded. See the actions of the **Discard** button.
- How images are handled is described in the section Images in the DB.

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 data 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.



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 *(My) Documents* MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\DBAndCharts**.

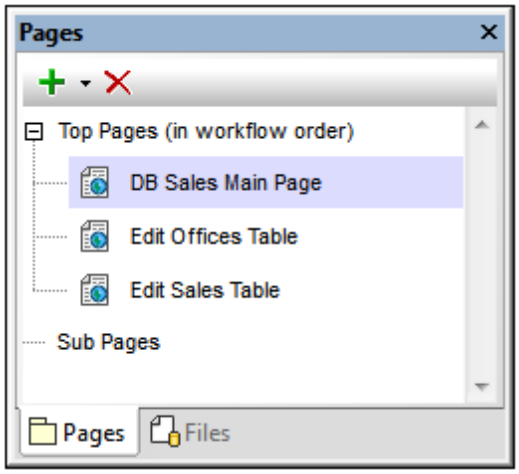
- The design file is **DBAndCharts.mtd**. Open it and read the descriptions in the tutorial to see how the design was built and how it works.
- The MS Access database, **OfficeSales_DB.mdb**, contains the tables used as the data sources of the design.

The Project Structure

Open **DBAndCharts.mtd** 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, Data Sources of the Main Page, for more information.)

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

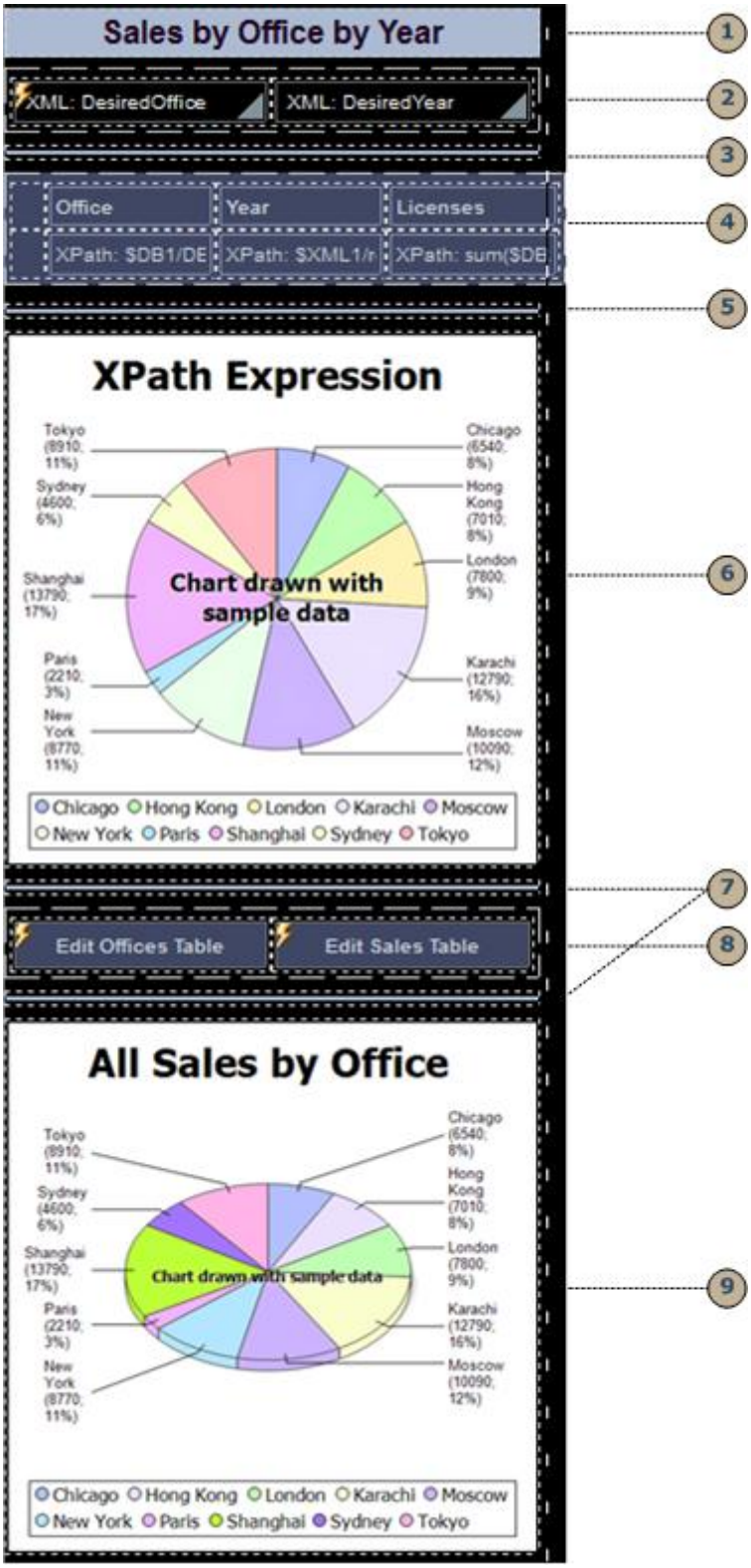
- *DB Sales Main Page:* This is the start page. It displays the DB data and has two buttons that go, respectively, to the other two top pages.
- *Edit Offices Table:* Is arrived at via a button click from the main page and enables editing of the DB's Offices table.
- *Edit Sales Table:* Is arrived at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

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.



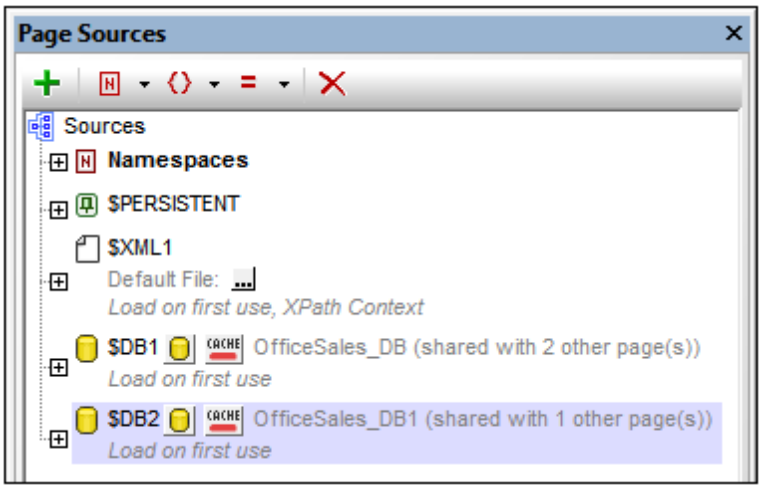
All the components are controls that have been dragged from the Controls Pane and dropped into the design. Each has then been assigned properties in the Styles & Properties Pane. For controls that need to be associated with data from the data sources, the appropriate source node has been assigned by dragging the data 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*
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*
6 9	Chart controls that display DB data in the form of charts. *See detailed description*
8	Button controls with `OnButtonClicked` actions that go to *Edit Offices* and *Edit Sales* pages

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

Data Sources of the Main Page

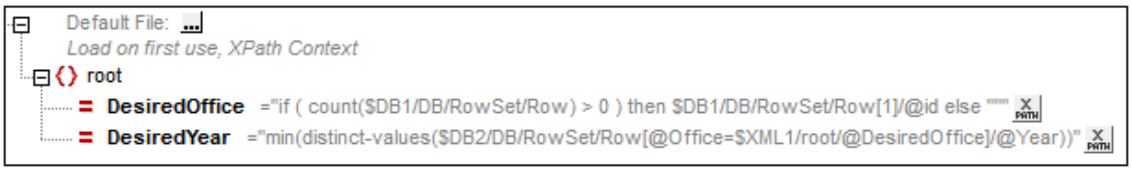
The main page has three data sources: **$XML1**, **$DB1**, and **$DB2**. These are displayed and managed in the Page Sources Pane (*see screenshot below*).



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 data source: $XML1

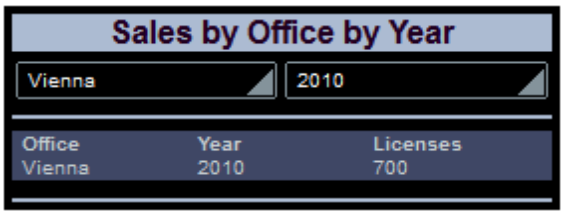
This data source was created as an editable empty XML. The root node **$XML1** contains a root element (**root**), which has two attributes (**DesiredOffice** and **DesiredYear**). The root node, `$XML1`, was set (via its context menu) as the XPath context node for the page two. No default file is set,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



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

- 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*).

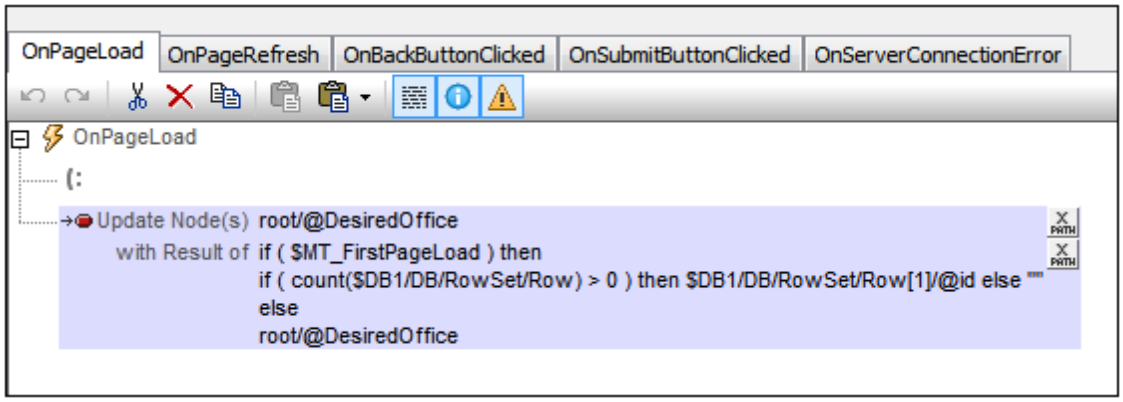


Each of the nodes has been given an initial value when the page loads (via the context menu command **Ensure Exists before Page Load (XPath Value)**). This is because the value of the node appears in the associated combo box, and we want the combo box to have an initial selection (*see simulator screenshot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

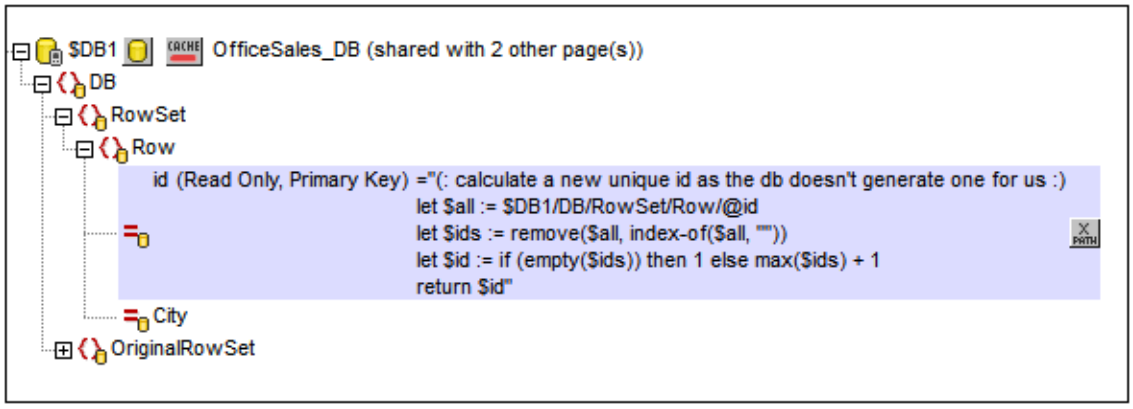
Additionally, we have specified that the **`@DesiredOffice`** node is correctly filled whenever the main page loads. This is done with an *Update Node* action on the **`OnPageLoad`** event of the main page (**Page | Page Actions**).



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 data source: $DB1

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



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 before Page Load (XPath Value)**:

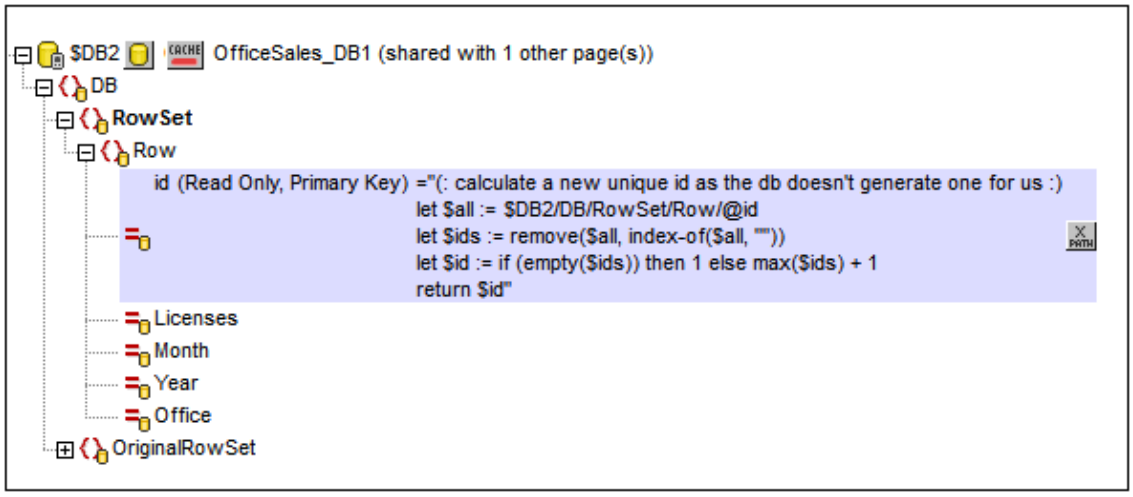
```
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```

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 data 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 data source: $DB2

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



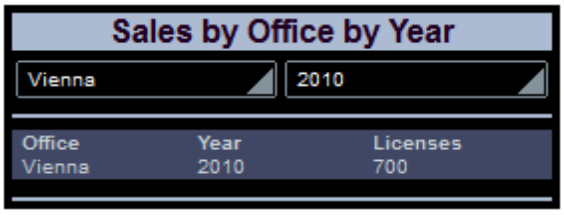
Each row in the Sales table has five columns (**id** , **Licenses**, **Month**, **Year**, and **Office**). The DB table row corresponds to the **Row** element in the data source tree. The table's columns correspond to the attributes of the **Row** element. The `id` attribute has an XQuery expression 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
```

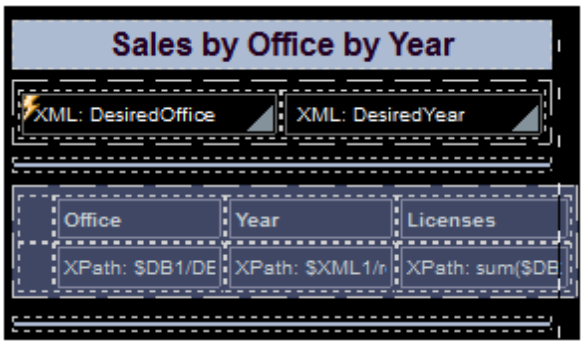
An **OriginalRowSet** node must be created (via the context menu) if any node in the data source is to be edited. This is required so that `OriginalRowSet` holds the original data while **RowSet** holds the current (edited) data.

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.



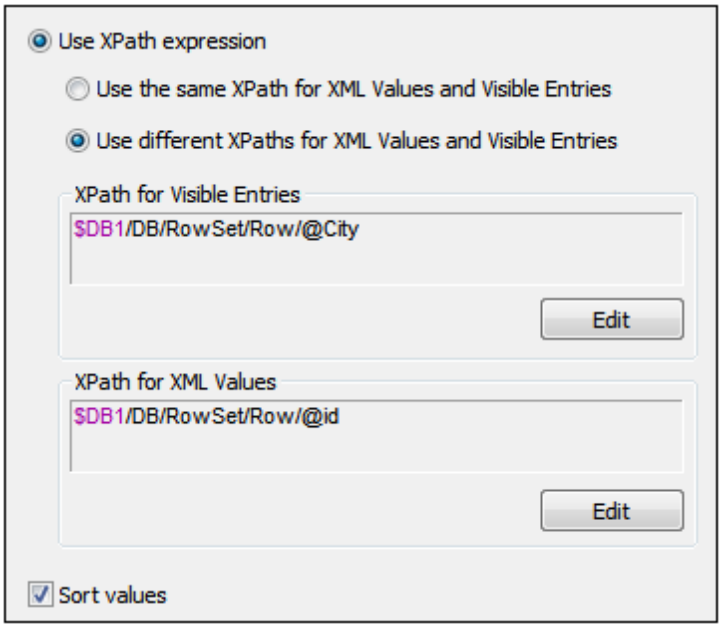
The next screenshot shows the combo boxes in the design. The combo boxes have been placed in separate cells of a table for layout purposes.



The DesiredOffice combo box

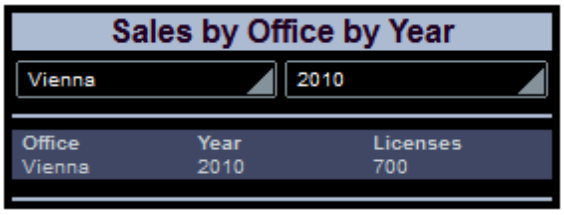
The following settings have been made:

- 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).



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.

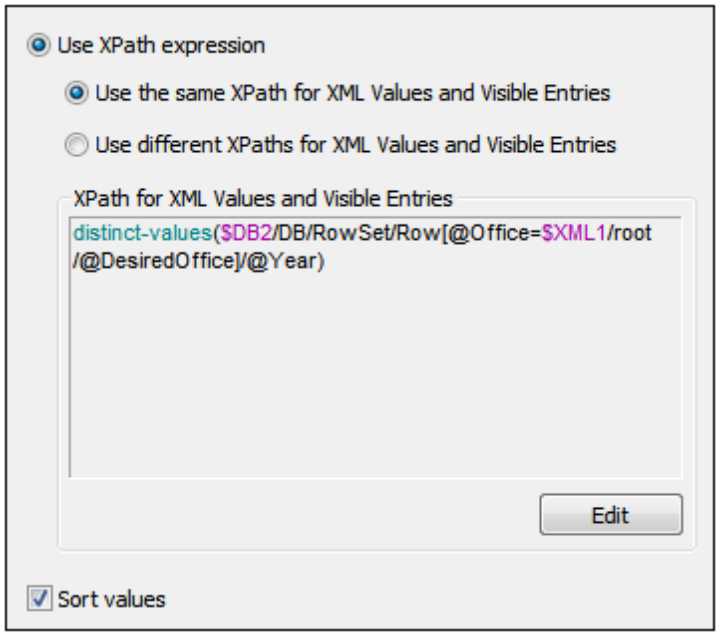


So, when Vienna is selected in the first combo box (as in the screenshot above), all the records in **$DB2** with `@Office='Vienna'` were searched and a sequence of the unique years in these records was created. The year with the minimum numerical value—in this case **2010**—is passed to the node to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

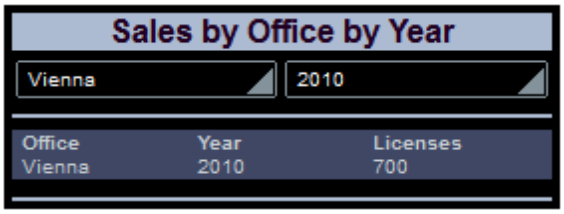
- A source node link is made between the combo box and the `$XML1/root/@DesiredYear` node by dragging the node onto the combo box. This serves to pass the combo box selection to the node and the value of the node to the combo box.
- The items in the dropdown list of the combo box are defined in the Edit 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).



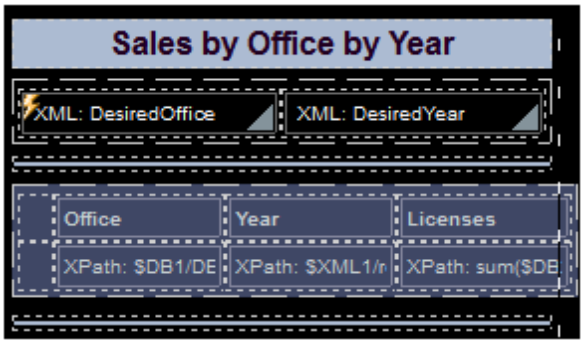
Notice that both the values in the dropdown list as well as those that will be passed to the XML node are the same. They are the sequence of all the unique years in which the selected office has recorded sales. Since the *Sort Values* check box has been selected the items of the dropdown list will be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

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.



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.



The XPath expressions are as follows:

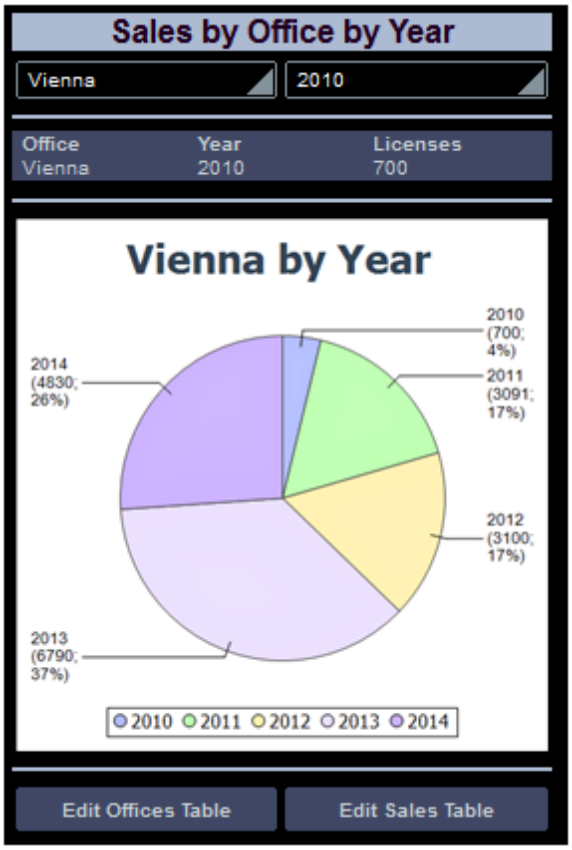
- *DesiredOffice:* Is taken from **$DB1**. It is the **@City** value of the **Row** with an **@id** equal to the **id** value of the combo box selection.
  $DB1/DB/RowSet/Row[@id=$XML1/root/@DesiredOffice]/@City

- *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)

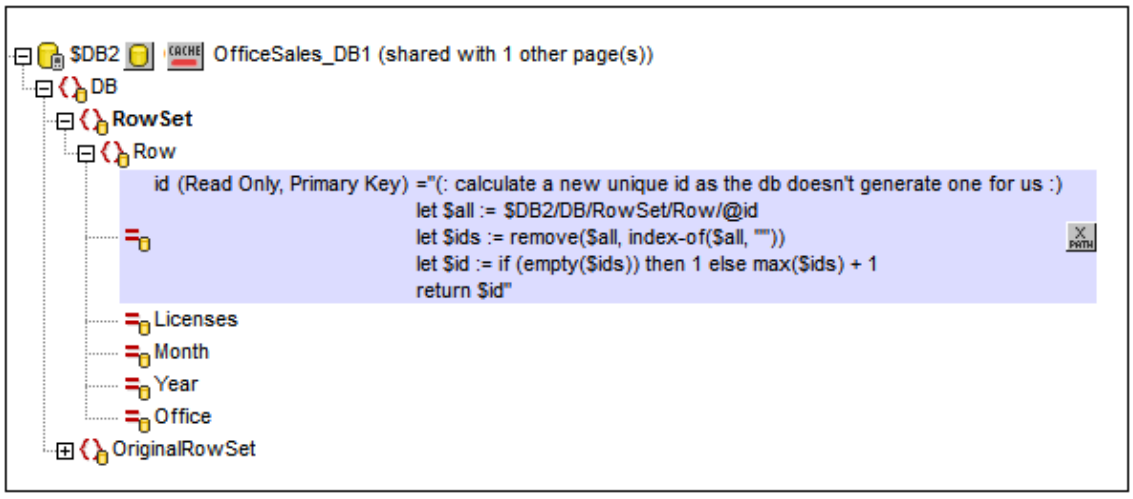
The Charts

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*).



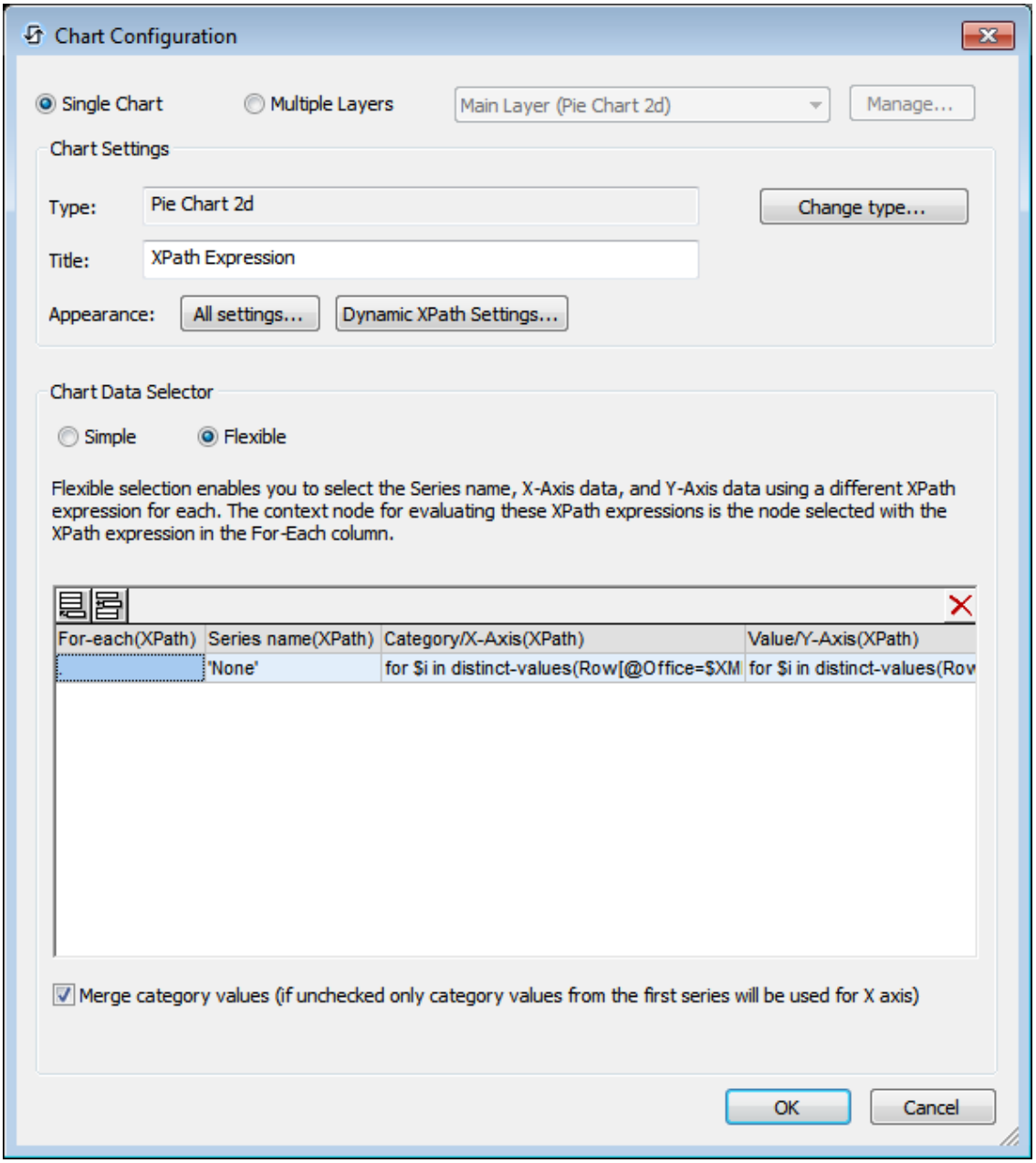
XPath context node

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.



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). Note that the chart type is a pie chart.



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
```

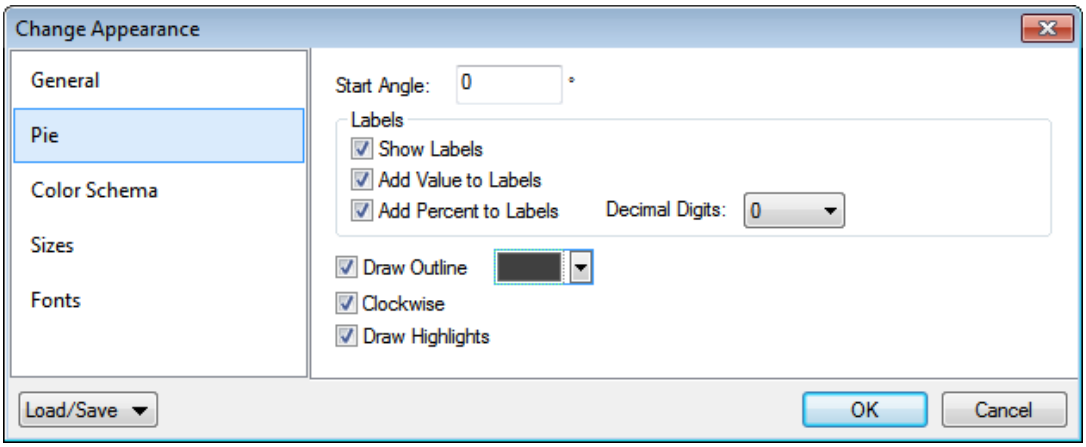
- *Y-Axis:* For the selected office and for each of its unique years, sums up the sales (stored in its `@Licenses` attribute)

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

Additional definitions

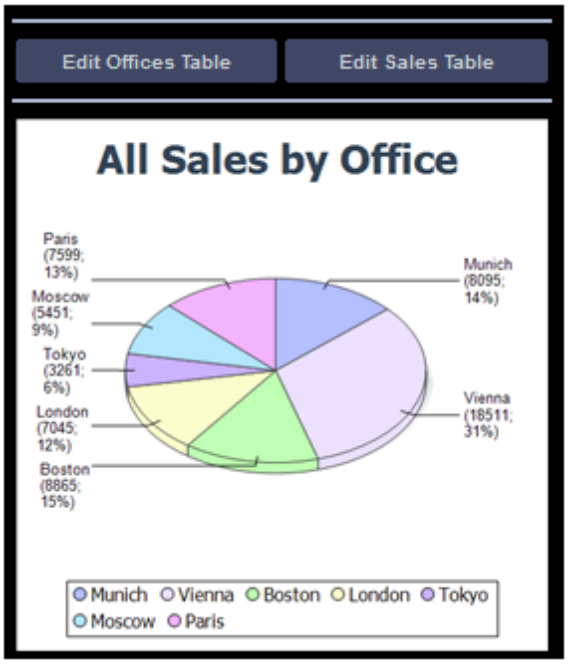
Additionally, the following settings were made:

- 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*.



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.



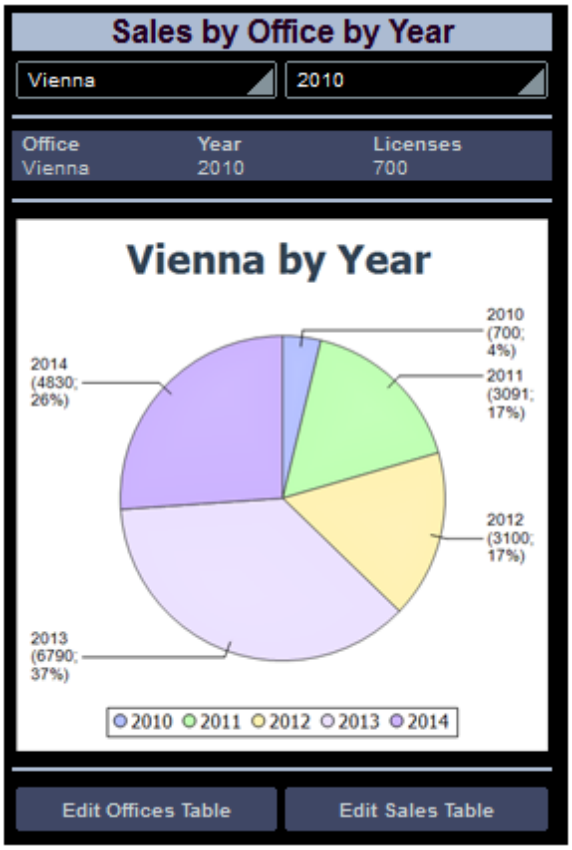
The XPath expressions are as follows:

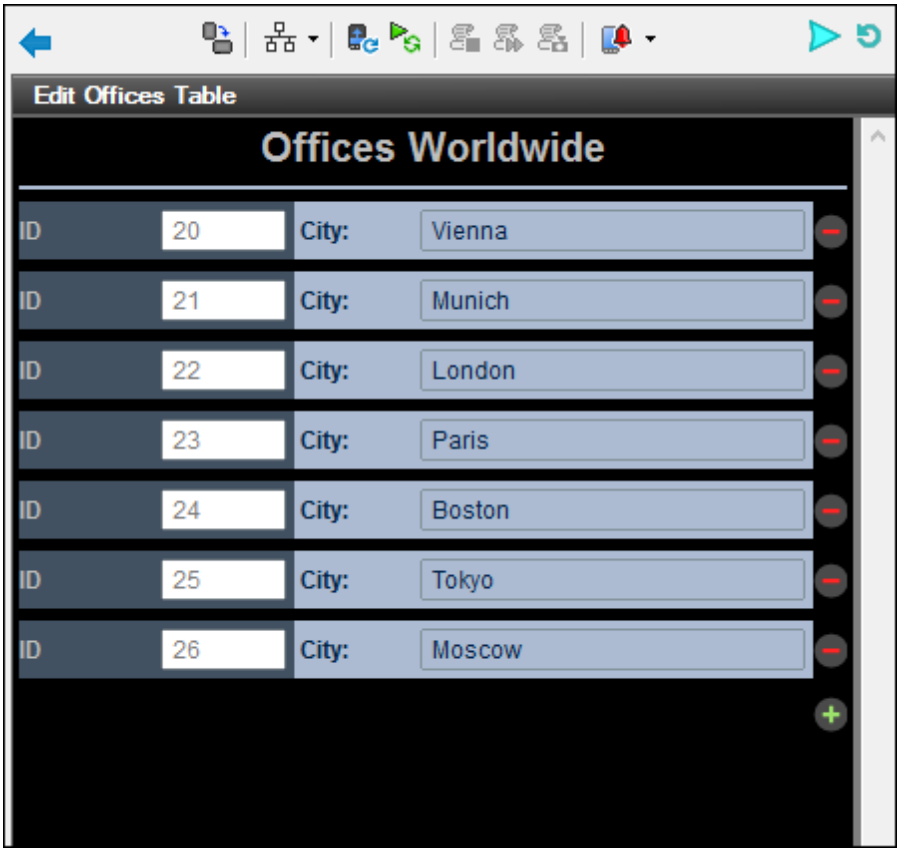
- *X-Axis:* Creates a sequence of the city names of offices (not IDs), with city names being taken from **$DB1**.
  ```
  for $i in distinct-values(Row/@Office) return $DB1/DB/RowSet/Row[@id=$i]/@City
  ```

- *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)
  ```

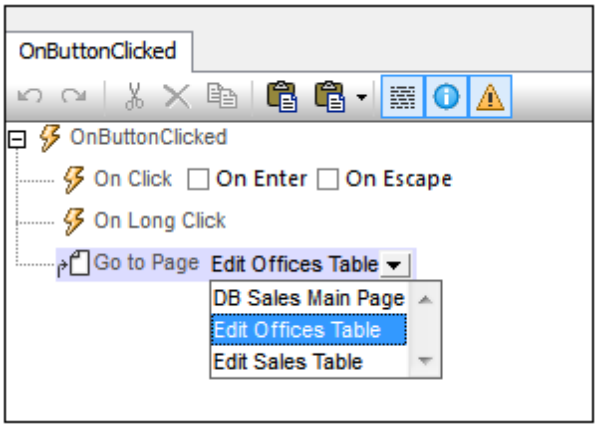
Edit Offices Table

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).





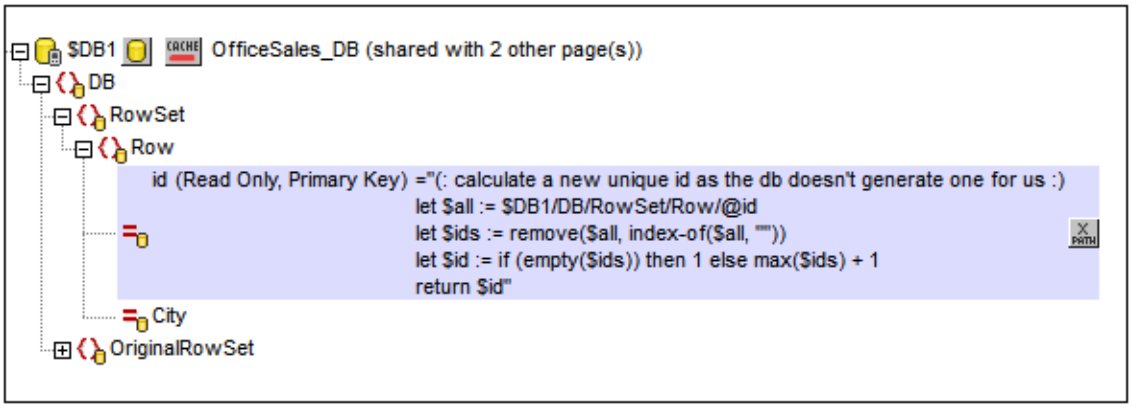
In the design, the **Edit** buttons (*first screenshot below*) have both been assigned the *Go to Page* action on their respective `OnButtonClicked` events (right-click the button and select **Control Actions for OnButtonClicked**). These *Go to Page* actions (*second screenshot below*) load the respective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

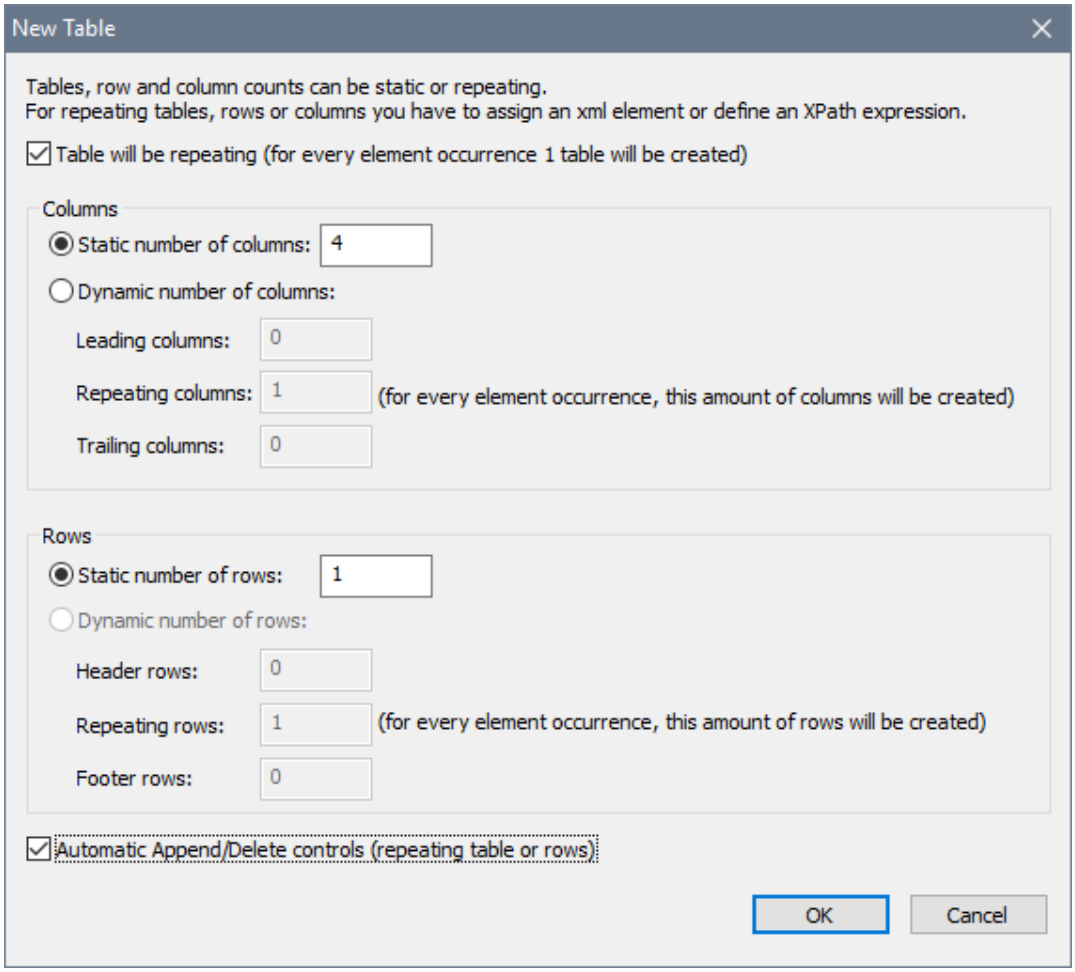


In the design, we will do the following:

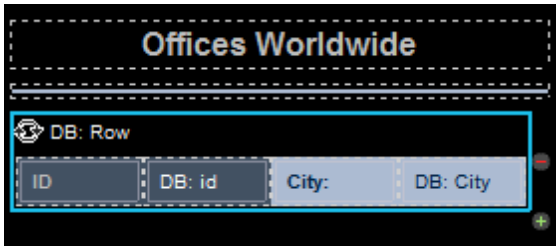
To...	How
Display all (`Office`) rows	Add a repeating table, with the `Office` row as the repeating element
Include controls for deletion and addition of rows	When adding the table, enable the automatic inclusion of Delete/ Append controls
Enable editing of `@City` values	Add an Edit Field control that has a source node to `@City`
Save changes back to DB	Add a *Save* action to the page's `OnSubmitButtonClicked` event; 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 and dropping it in the design, the New table dialog appears (*screenshot below*).

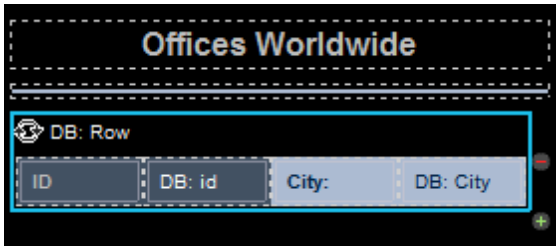


Specify that the table will be repeating, enter the number of columns `(4)` and rows `(1)`, select the *Automatic Append/Delete Controls* check box, and click **OK**. Labels are added to the first three cells of the row, as shown in the screenshot below. A source node link to the `@id` node of `$DB1`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



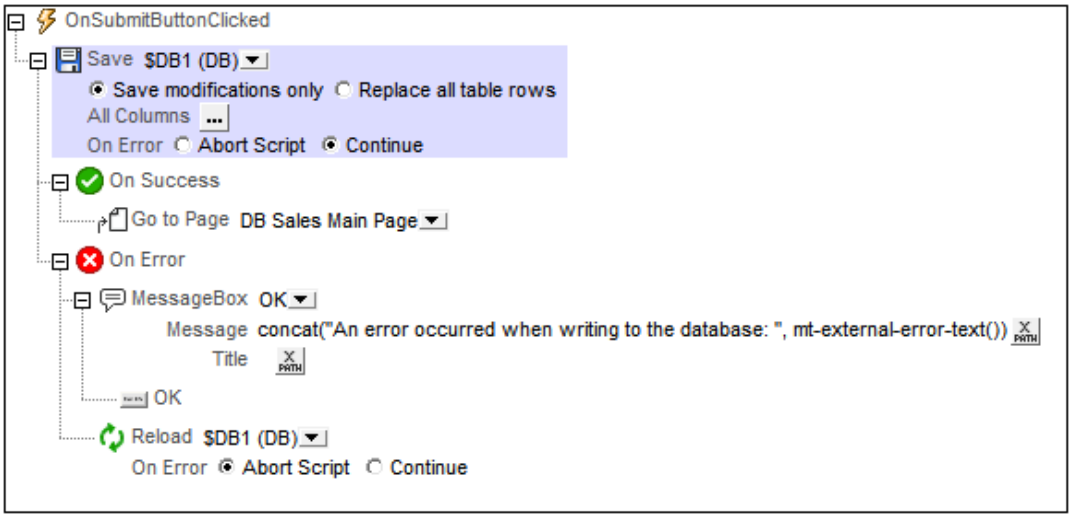
- Enable editing of @City in $DB1

  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.



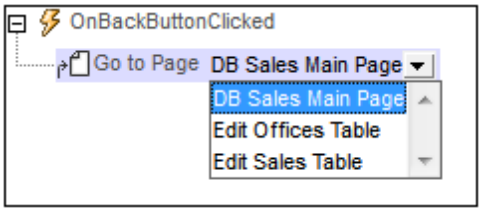
- Page actions: 'Save' and 'Go to page'

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



  Actions are defined for the following events:

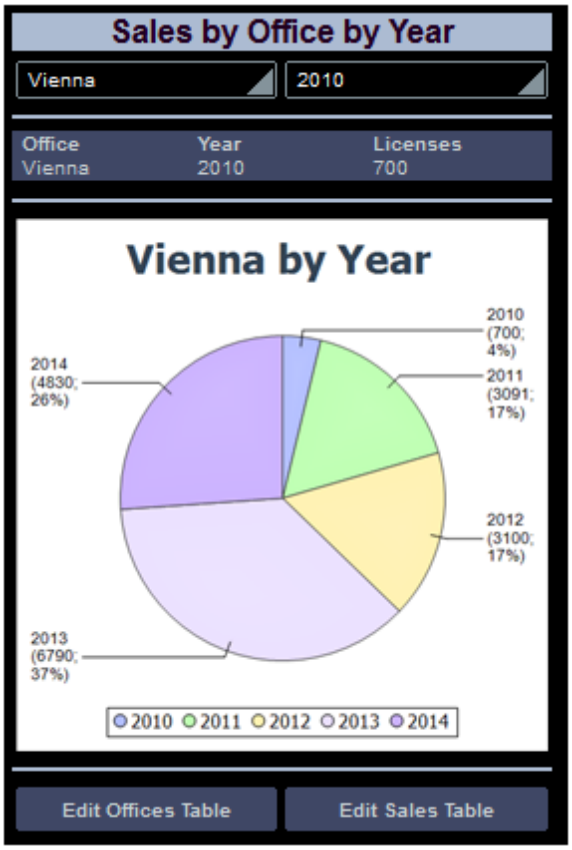
  - **`OnSubmitButtonClicked`**: Saves all columns of the page to the DB (`$DB1`) and goes back to the main page. You might also want to add the Reload action so that the DB is reloaded with the unmodified data in case the record is not saved to the DB (*see screenshot above*).
  - **`OnBackButtonClicked`**: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



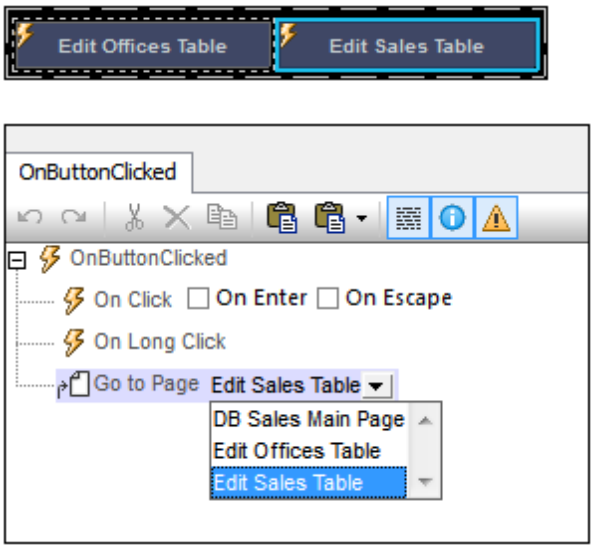
  The tree of the data source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. Original data is saved in the `OriginalRowSet` element, so that the columns of the `RowSet` element can be edited. The `OriginalRowSet` element is updated with the new value only when the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

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).



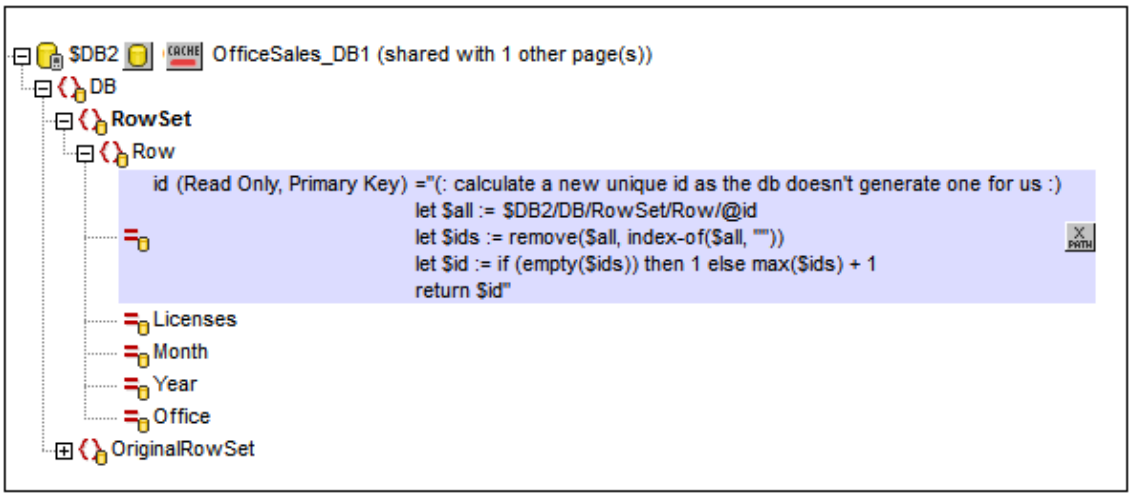
In the design, the **Edit** buttons (*first screenshot below*) have both been assigned the *Go to Page* action on their respective `OnButtonClicked` events (right-click the button and select **Control Actions for OnButtonClicked**). These *Go to Page* actions (*second screenshot below*) load the respective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

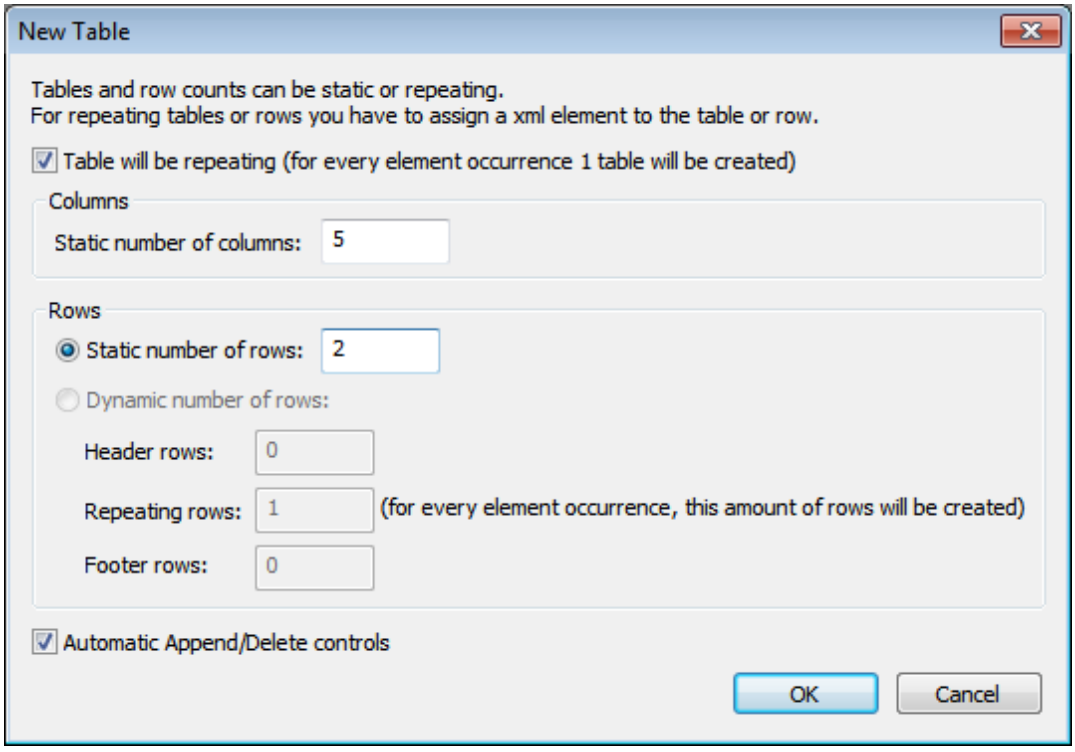


In the design, we will do the following:

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

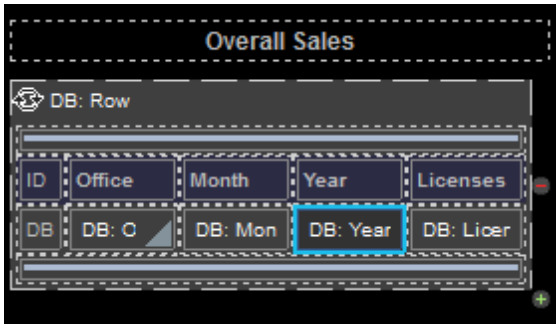
- Add a repeating table that has Append/Delete controls

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



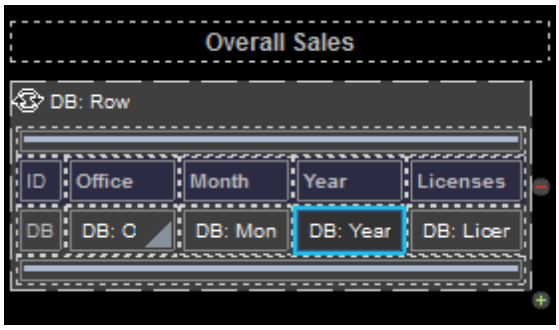
  Specify that the table will be repeating, enter the number of columns `(5)` and rows `(2)`, select the *Automatic Append/Delete Controls* check box, and click **OK**. Labels are added for headers to the cells of the first row. A label is added for the uneditable `@id` value to the first cell of the second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

(`DB:id`).



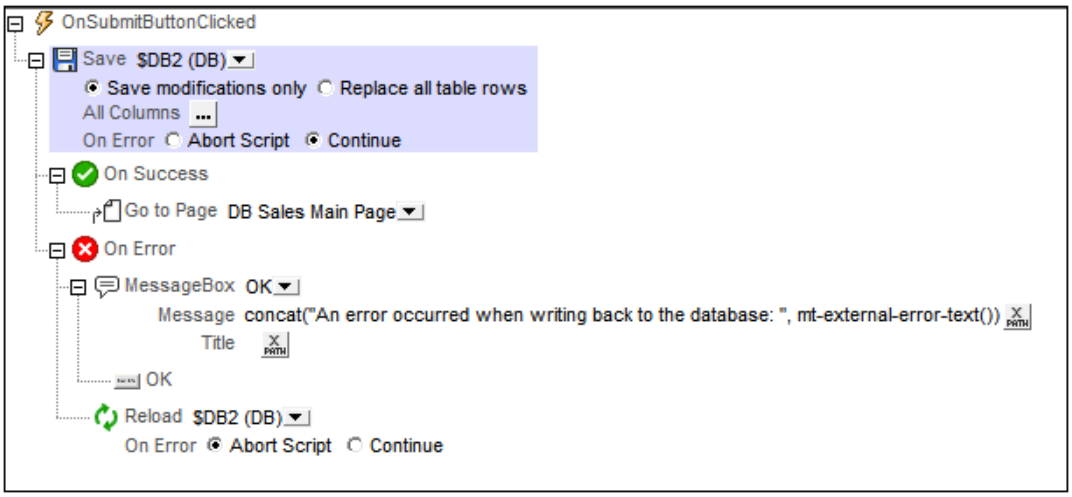
- Enable editing of the editable nodes

  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.



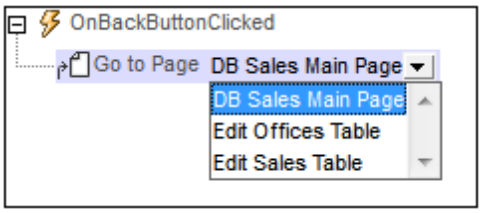
- Page actions: 'Save' and 'Go to page'

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



Actions are defined for the following events:

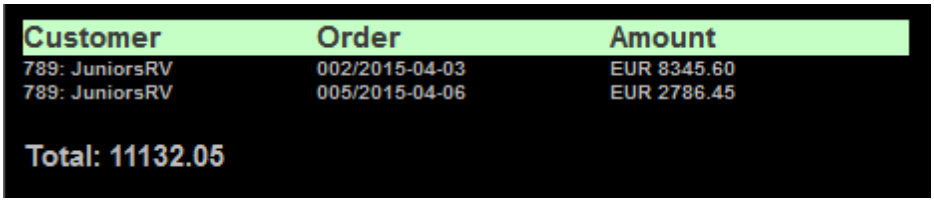
- **`OnSubmitButtonClicked`**: Saves all columns of the page to the DB (`$DB1`) and goes back to the main page. You might also want to add the Reload action so that the DB is reloaded with the unmodified data in case the record is not saved to the DB (*see screenshot above*).
- **`OnBackButtonClicked`**: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



The tree of the data 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*).



The data for the design is stored in two data sources: one that stores customer data, and a second that stores order details. The two data 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 data 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 *(My) Documents* 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
- Data Source Listings
- Top Page: Data Sources

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

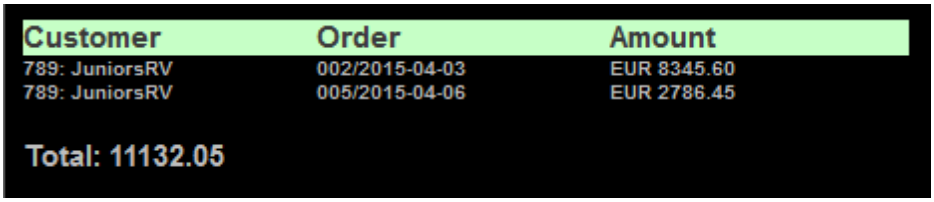
- Create the top page and two data sources: **$XML1** and **$CUSTOMERS**
- Create a dynamic table to contain customer data from $CUSTOMERS. Each row of the table will correspond to one customer in the XML data source $CUSTOMERS
- Create a sub page named Orders
- Create an action group that does the following: (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 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. 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

Sub page: Orders

- Create the three data sources for the sub page: **$XML1** (shared with top page), **$CUSTOMERS** (shared with top page), and **$ORDERS**
- Create a dynamic table to display the order details in the data file (*screenshots below show the order tables of (i) a selected customer, and (ii) all customers*). Each row of the

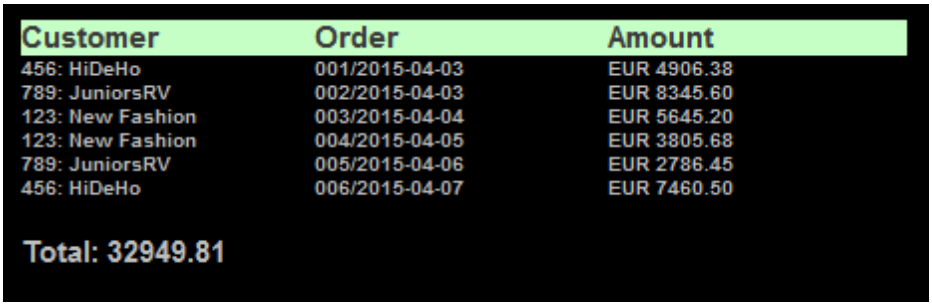
table will correspond to one order in the XML data source $ORDERS

- Set up the visibility property of the table's repeating row group to show (i) only the customer selected on the top page, or (ii) all customers
- Create an XPath expression to generate the total amount 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

Data Source Listings

The design will have three XML data 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 $XML1 data source is shared by both pages of the design.
- **$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*). The $CUSTOMERS data source is used in both the top page as well as the sub page.
- **$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*). The orders in the $ORDERS data source are displayed in a table in the Orders sub page.

Listing of the XML data source, Customers.xml

Located in the MobileTogether folder of the *(My) Documents folder*:
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 data source, Orders.xml

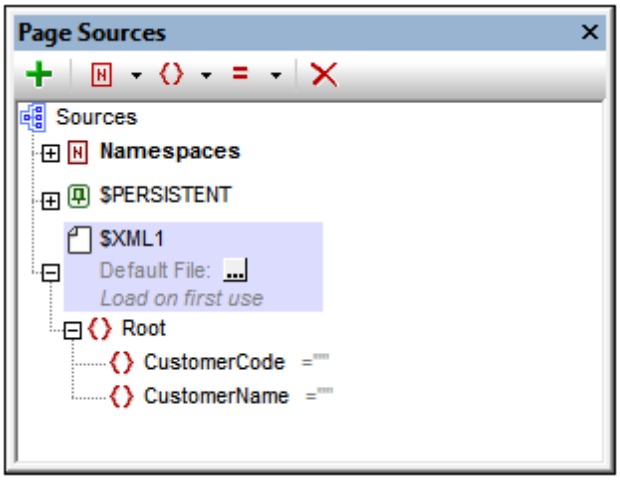
Located in the MobileTogether folder of the *(My) Documents folder*:
MobileTogetherDesignerExamples\Tutorials\AltovaProducts.xml.

```
<?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>
```

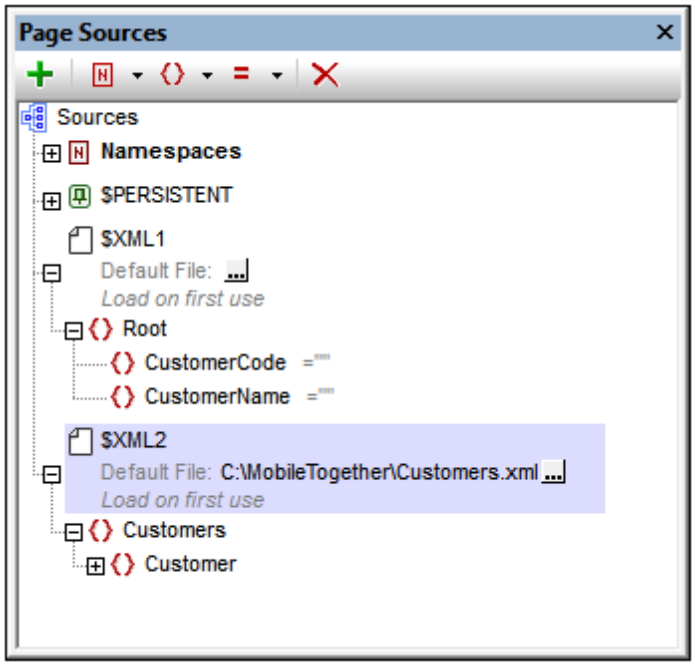
Top Page: Data Sources

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

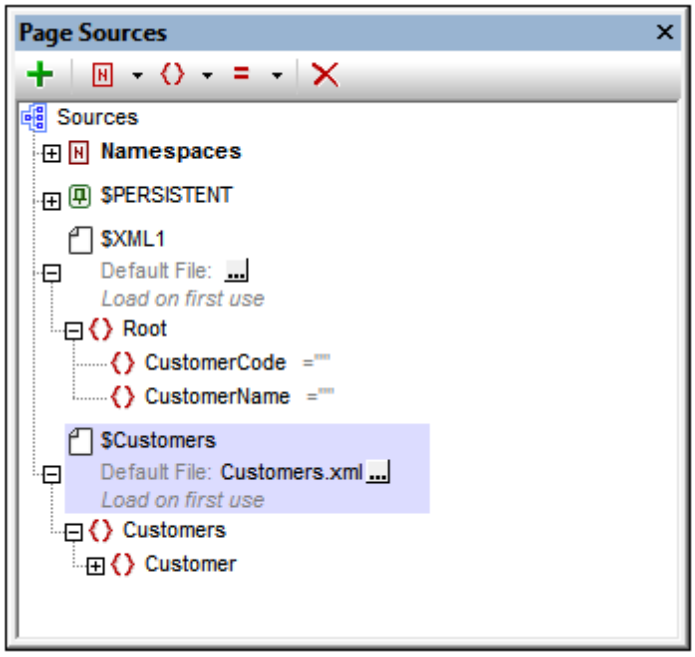
1. Create a new design file with the **File | New** command.
2. Save the file with any name you like.
3. In the Pages Pane, rename the top page (which is created by default) to `Customers`. Do this by double-clicking the page name and then editing it.
4. In the Page Sources Pane, add a new empty XML source by clicking the **Add Source** icon, and selecting *New, empty XML*. In the second screen, keep the default selections. A data source called `$XML1` will be created (*see screenshot below*).
5. Manually build the structure of this data source so that it is as shown in the screenshot below. Do this by right-clicking nodes in the tree (starting with `$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)**. 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.



   At runtime, we plan to have the nodes of the `$XML1` tree updated with the data (code and name) of the customer that the end user selects from the list of customers displayed on this page.
7. Create a second data source by clicking the **Add Source** icon, and selecting *New XML or HTML structure imported from file*. Browse for the file `Customers.xml` and click **Open**. When you are prompted about whether to deploy the file, choose **Yes**. A data source called `$XML2` will be created (*see screenshot below*).



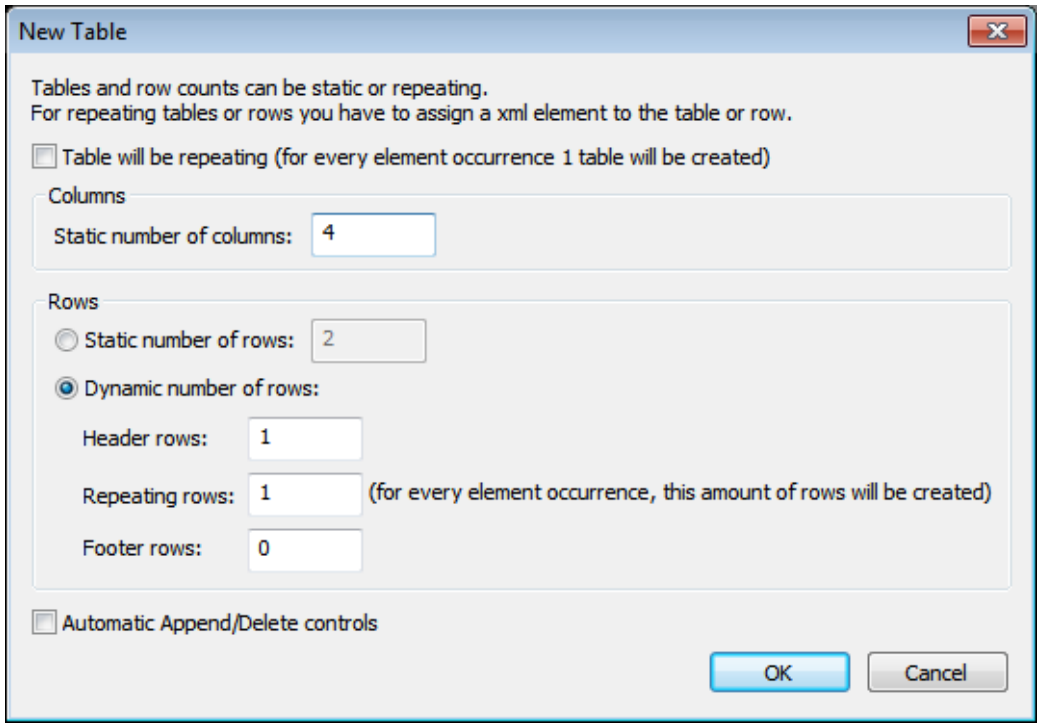
8. Double-click the root node `$XML2` and edit the name to `$CUSTOMERS` (*see screenshot below*).
9. Click the **Additional Dialog** button of the `$CUSTOMERS` default file. In the dialog that appears, select the *Relative paths* check box to make the file's path relative to the design (*see screenshot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



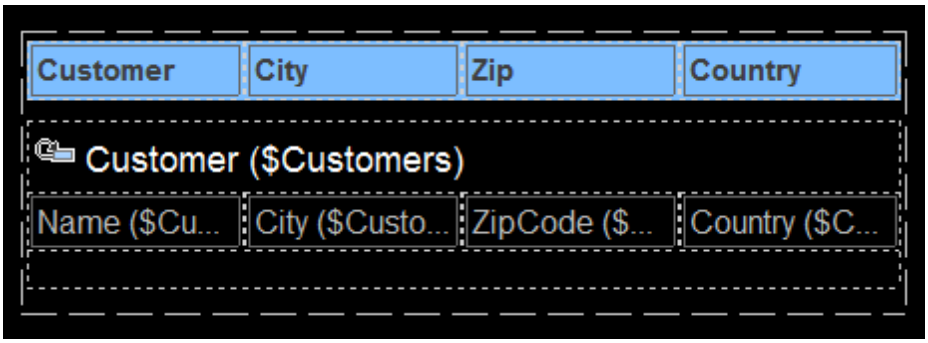
Top Page: Customers Table

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

1. Drag a Table control from the Controls Pane and drop it into the design.
2. In the New Table dialog that appears (*see screenshot below*), create the table as a dynamic table. 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 data source. Specify that the table has four columns and one header row (*see screenshot below*). Click **OK** to create the table.



3. From the Page Sources Pane, 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 from the Controls Pane 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).
6. Drag a Label control from the Controls Pane and drop it into the first column of the table-body row. Then, from the Page Sources Pane, drag the `Customer/Name` element of the `$CUSTOMERS` data 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*).



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

Top Page: Action Group, Go to Sub Page

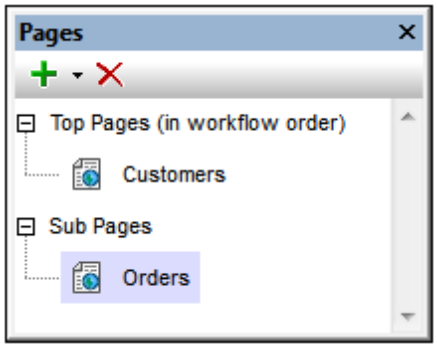
In the previous section, we created a table that displays the details of each customer in a separate table row. When the end user clicks on any customer detail (say name or city), we want to display that customer's current orders. We plan to do this by creating a sub page that will filter all current 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` data 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, and then assign this action group to each label's `OnLabelClicked` event. Before we define the sequence of actions we need to create the sub page that the Go to Subpage action will target.

Create the sub page

Click any item in the Pages Pane 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

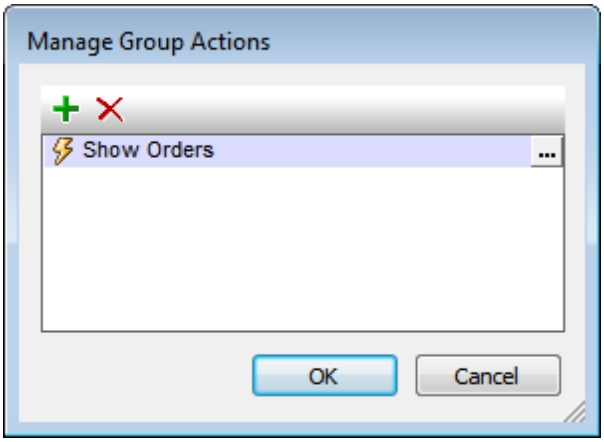
We will create an action group consisting of the following actions:

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

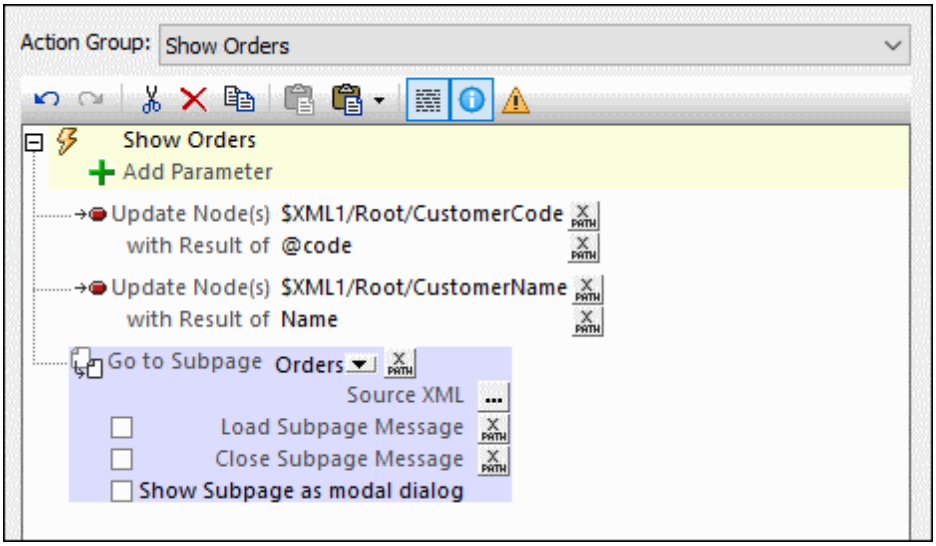
After we create the action group, we can assign it to the `OnLabelClicked` event of each of the four table-body-row labels. At runtime, the action sequence will be executed when the end user clicks the label of any customer detail.

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 that appears, in the left-hand pane, click the **Manage** button of Action Groups.
3. In the Manage Group Actions dialog that appears (*screenshot below*), click the **Add a Group** icon in the menu bar. Rename the added group to Show Orders (*screenshot below*; you can double-click the name to edit it).



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



6. Click **OK** when done.

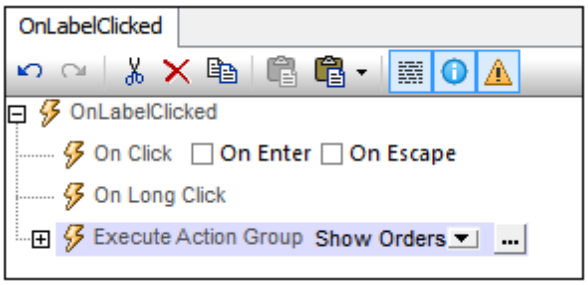
We have set the following updates: (i) the `$XML1//CustomerCode` element will be updated with

the `Customer/@code` value of the selected customer (from `Customers.xml`), and (ii) the `$XML1//CustomerName` element will be updated with the `Customer/Name` value of the selected customer (from `Customers.xml`).

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 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.



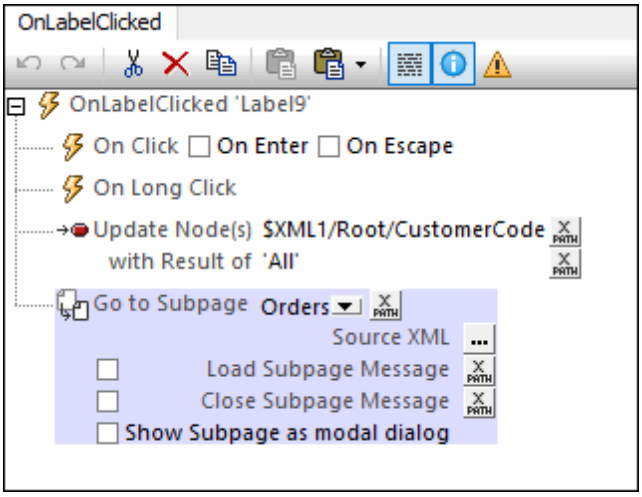
3. Repeat Step 2 for each of the three other labels of the table-body row. This ensures that the Show Orders sequence of actions will be executed when any of the four labels of a row is clicked.

Top Page: Show All Orders Action

In the previous section, 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` database.

Add the *Show all orders* label as follows:

1. Drag a Label control from the Controls Pane 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.
3. Open the Actions dialog via the control's context menu command **Control Actions OnLabelClicked Action**, and add a sequence of actions for the `OnLabelClicked` event (*see screenshot below*).



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.

Sub Page: Data Sources

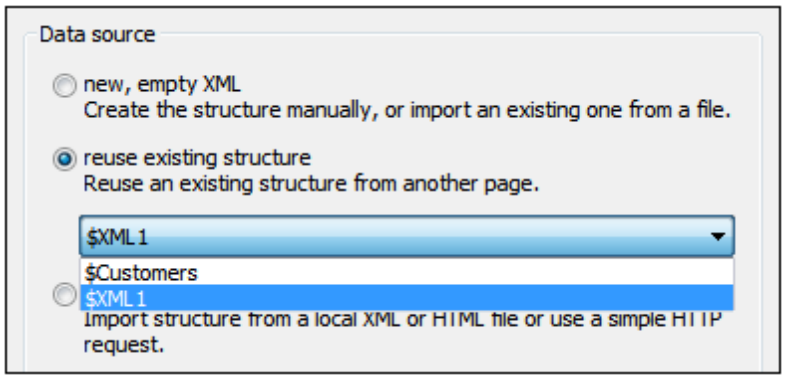
The sub page has already been created because this step was needed earlier, in order to define the Go to Subpage action. Our sub page is called Orders. It will have the following data sources:

- **$ORDERS**, which will have the structure and content of `Orders.xml`. This data source is needed in order to display the orders contained in `Orders.xml`
- **$XML1**: This data 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 data 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.

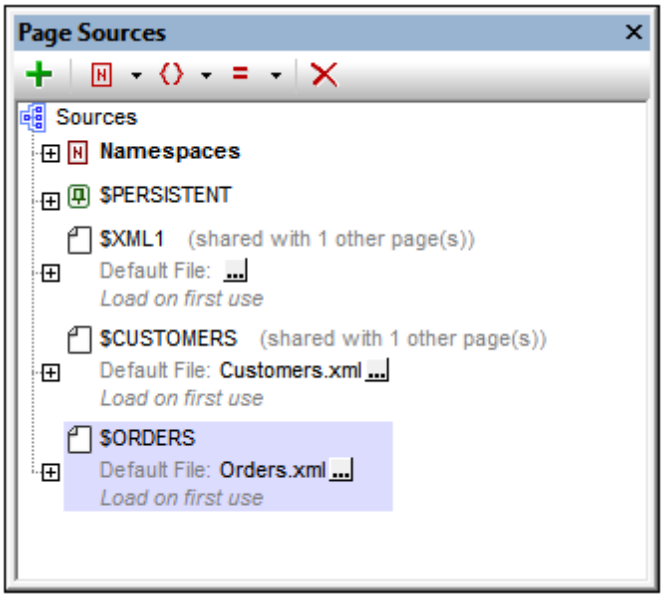
Adding the data sources

Add the three data sources as follows:

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



   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 data source, `$CUSTOMERS`, in the same way. Note that the structure, content, and default file will be as created in the top page.
4. In the Page Sources Pane, click the **Add Source** icon, and select *New XML or HTML structure imported from file*. Browse for the file `Orders.xml` and click **Open**. When you are prompted about whether to deploy the file, choose **Yes**. A data source called `$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. 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**.

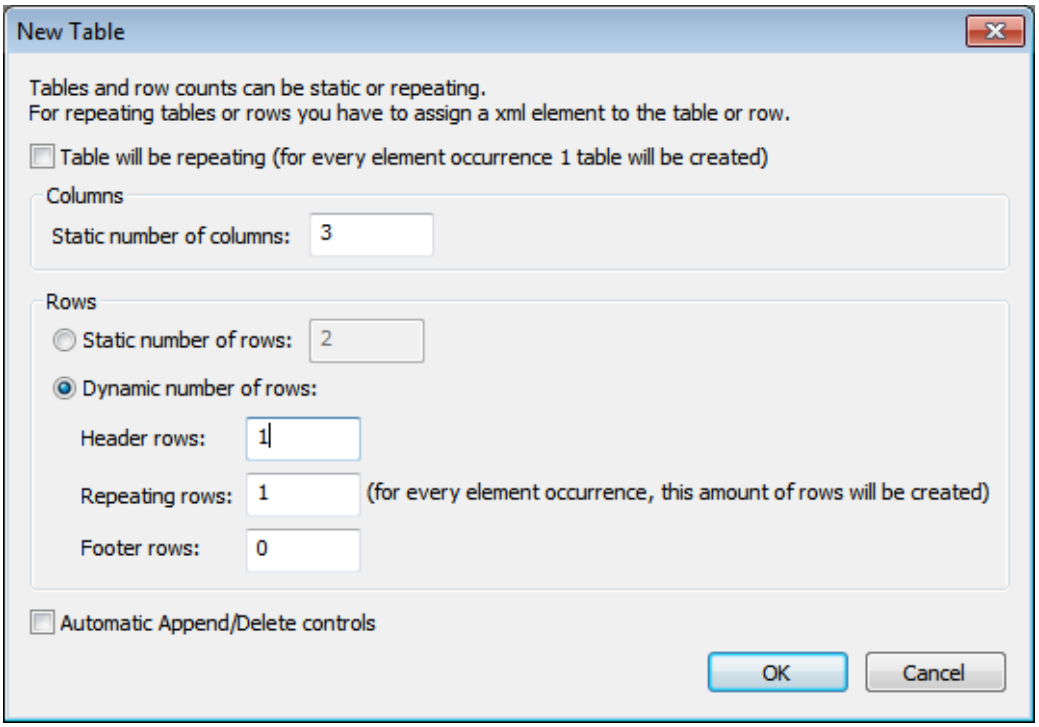


After the data sources have been added, you are ready to create the design of the sub page.

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 data source **`Orders.xml`**. Create the Orders table as follows.

1. Drag a Table control from the Controls Pane and drop it into the design.
2. In the New Table dialog that appears (*see screenshot below*), create the table as a dynamic table. 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 data source. Specify that the table has three columns and one header row (*see screenshot below*). Click **OK** to create the table.

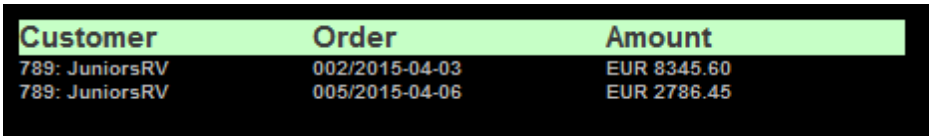


3. From the Page Sources Pane, drag the `Order` element to the **Repeating Row** icon of the table in the design. Each `order` element will now correspond to one row of the table, and the `order` element will be the XPath context node of the table.
4. Drag a Label control from the Controls Pane and drop 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.
6. Drag Label controls from the Controls Pane 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.

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, select the label's `Text` property and click the XPath icon in the pane's menu bar. In the XPath dialog 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. 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.

Sub Page: Visibility Property

The Orders table that we have created in the Orders sub page is a dynamic table that generates one row for each `Order` element (or record) in the data source `Orders.xml`. 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, 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 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:

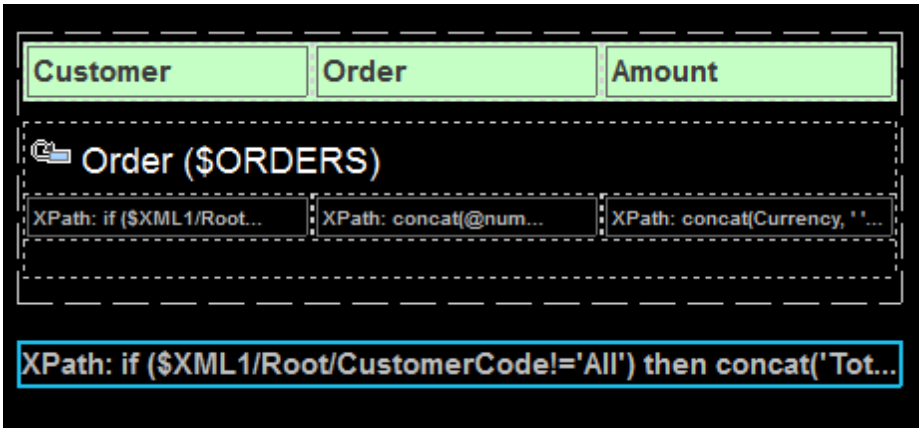
1. The `if` clause of the expression tests whether the element `$XML1/Root/CustomerCode` contains the string `All`.
2. If the element `$XML1/Root/CustomerCode` **does not contain** the string `All`, then all `Order` elements that have their `CustomerCode` element content equal to the content of the `$XML1/Root/CustomerCode` element will be selected. In effect, these will be the `Order` elements of the customer that was selected by the end user. Remember that the customer's `CustomerCode` has been stored in the `$XML1` data source (*see Top Page: Action Group, Go to Sub Page*).
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.

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 from the Controls Pane and drop it below the Orders table (*see screenshot below*).
2. In the Styles & Properties Pane, Click the **XPath** icon of the control's `Text` property.
3. In the In the Edit XPath/XQuery Expression dialog that appears, enter the XPath expression to calculate the total amounts (*the expression is given below*), and click **OK**.



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` data source (*see Top Page: Action Group, Go to Sub Page*).
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

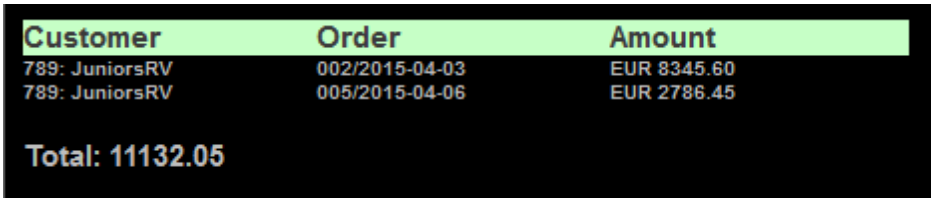
more than the two decimal places we require in a currency. We therefore use the `xs:decimal` type converter to round the `xs:double` to a two-decimal place number.

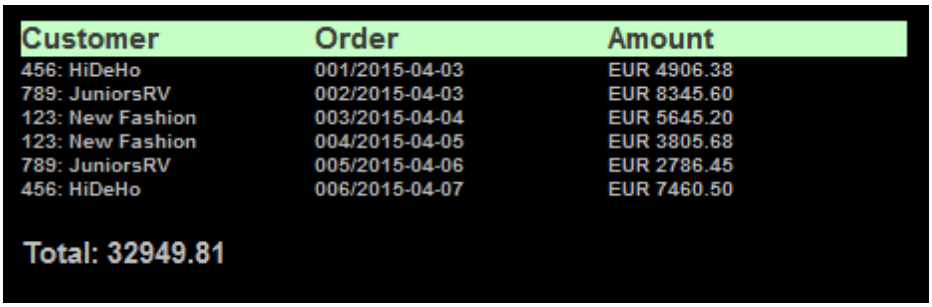
Simulation and Testing

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

Sub page: Orders (for selected customer and all customers, respectively)



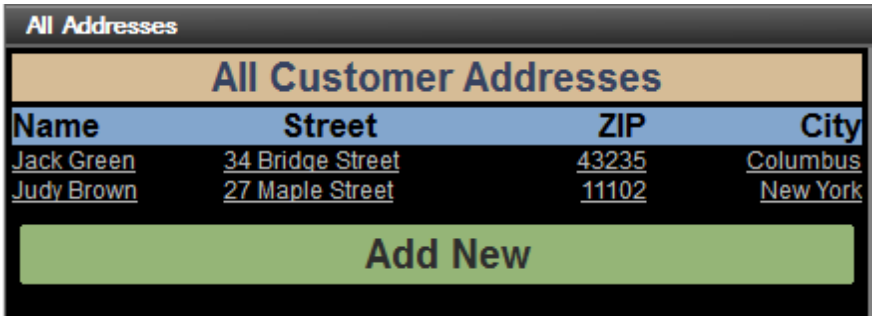


That's it!

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



Design plan

The design is constructed in this way:

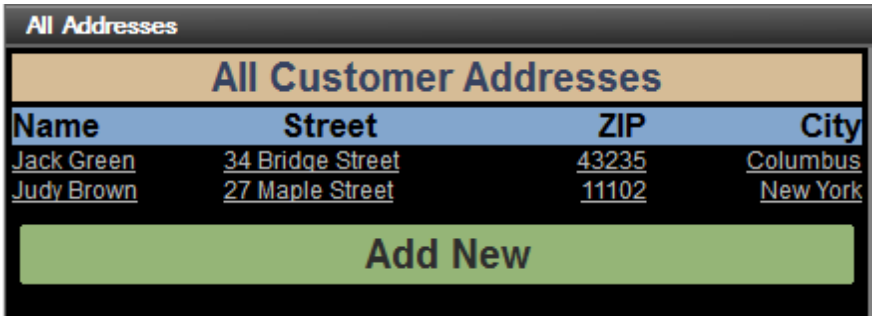
- Two XML data sources are used. The **$PERSISTENT** tree is used to hold the customer database. A **$EDIT** tree is used to hold the one record (a **Customer** element) that is being currently edited.
- The **$EDIT/Customer** element is created when an **Add New** button is clicked.
- When a **Save** button is clicked, the **$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

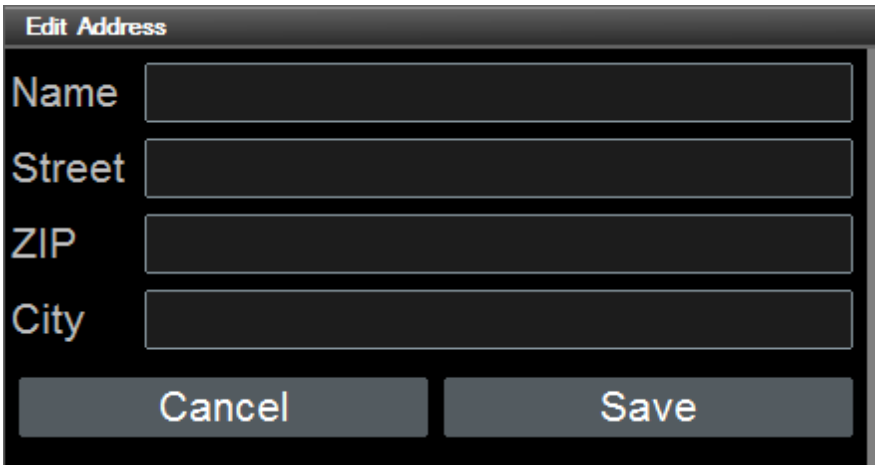
The finished tutorial file is located in your *(My) Documents* MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\AddEditRecords\AddAndEditRecords.mtd**. Open this file and check the design settings in it while reading through this tutorial.

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.



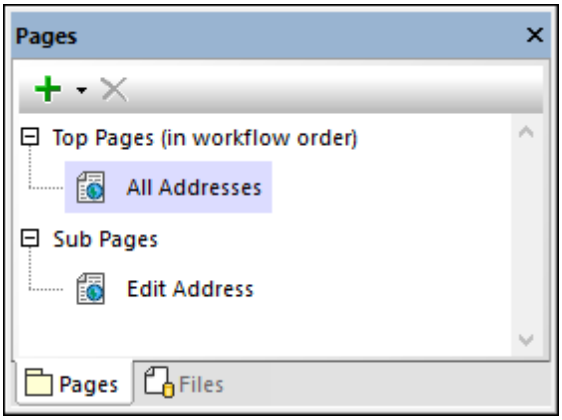
When the user clicks the **Add New** button, the button's action takes it to a sub page called *Edit Address (screenshot below)*. This page displays a data entry form in which the user can enter details of a new customer. The sub page is required so that the workflow can move between two cleanly separated 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.

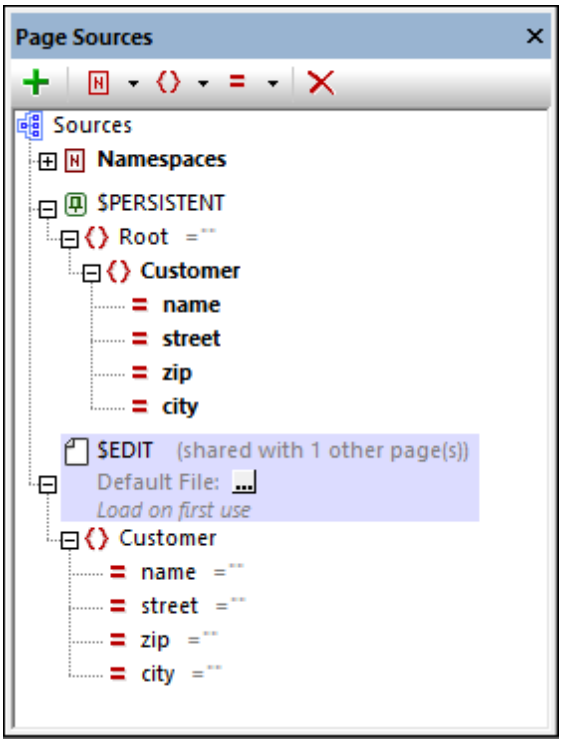
Creating the pages

The top page and sub page are created in the Pages pane (*screenshot below*).


Pages
Top Pages (in workflow order)
All Addresses
Sub Pages
Edit Address
Pages
Files

Data 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 or the context menus of nodes in the tree.



Note the following points:

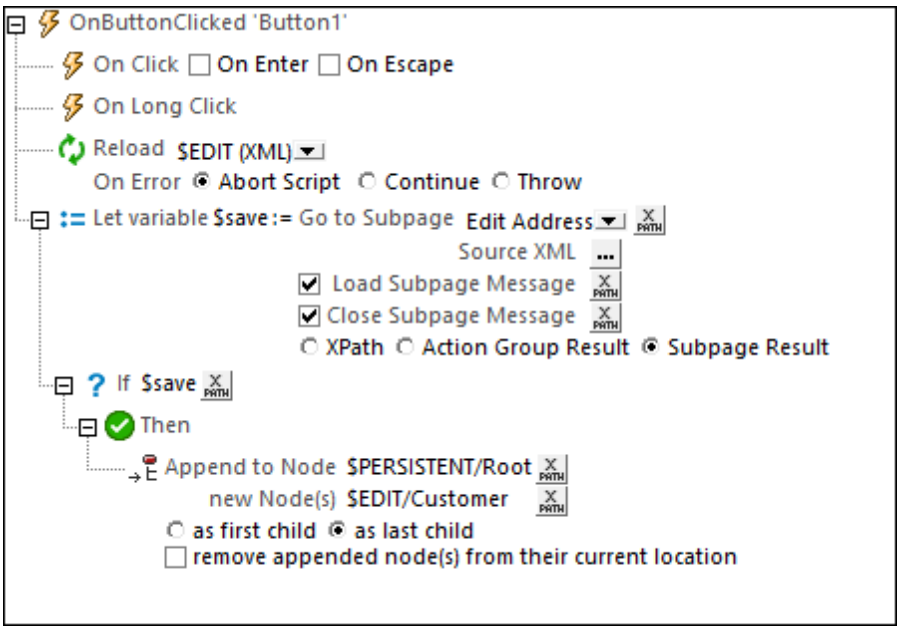
- In both trees, a **Customer** element corresponds to a single customer record.
- The data of each customer is stored in the attributes of that customer's **Customer** element.
- The **$EDIT** tree is shared between both pages of the design. This means that the data available in the tree is commonly available on both pages.
- In the **$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 in the design: they are used to display the customer database in the columns of the table on the main page (*see screenshot below*).


All Customer Addresses
Name
Street
ZIP
City
Customer ($PERSISTENT)
name ($PERSI...
street ($PERSI...
zip ($PERSIST...
city ($PERSIST...
Add New

Add a New Record

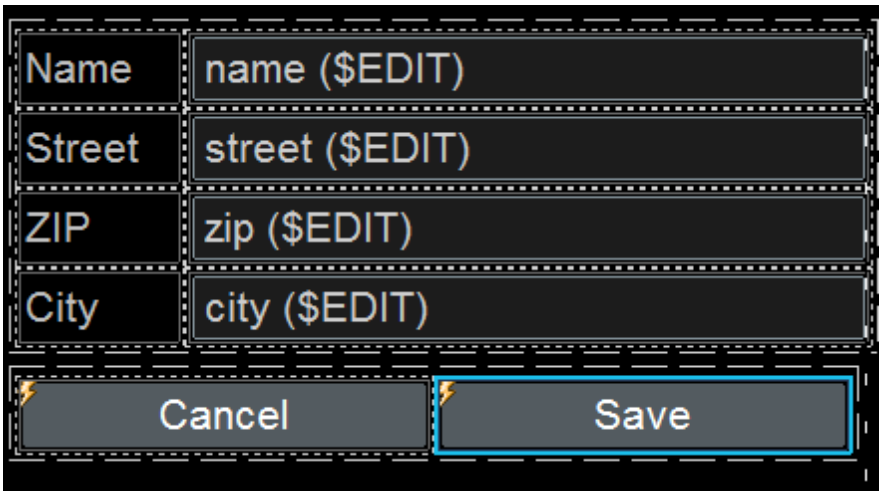
When the user opens the solution, the customer database is empty. To add a record, the **Add New** button (*see screenshot below*) is clicked.

The **Add New** button has two `onButtonClicked` actions (Reload and Let), which are shown in the screenshot below.



These two actions carry out the following sequence of steps:

1. The Reload action reloads the **$EDIT** tree. Since the nodes of this tree have been defined to be loaded with a fixed value of the empty string, all the fields of the customer record will be empty.
2. The Let action creates a variable called **$save**, which goes to the sub page *Edit Address* (*screenshot below*) and fetches its result.

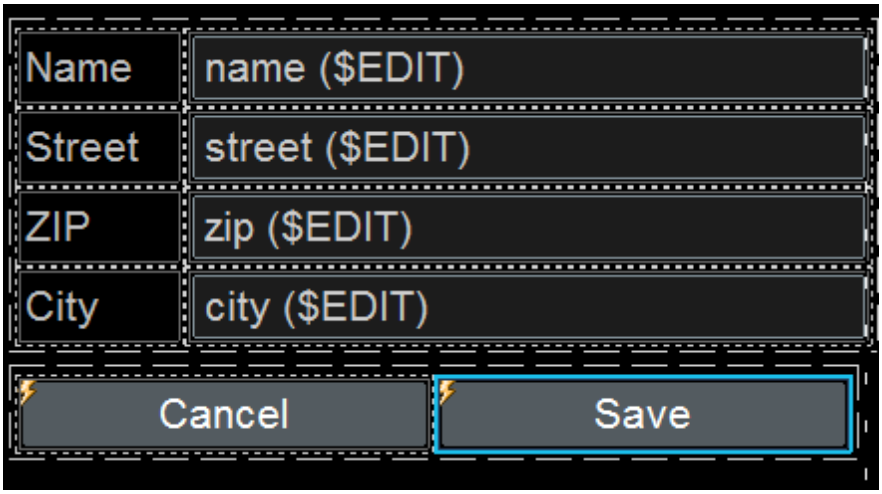


The result of the sub page is returned when the **Save** button of the page is clicked. The **Save** button executes the Close Subpage action and returns its result—which is the `Customer` node. So this node is then stored in the `$save` variable.

3. An If-Then action 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 `$EDIT/Customer` element to be appended (by using the Append Node(s) action) as the last child node of the `$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 without returning a result. The effect is to return to the main page without modifying the customer database in any way.

Enter New-Record Data

The *Edit Address* (sub) page (*screenshot below*) is used to enter new-customer data as described below:



- Each customer data edit field is associated with a page source link 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.
- The **Cancel** button closes the sub page without returning any result. This is defined via the Close Subpage action of the **Cancel** button.

Display All Records

The records of the customer database which is stored in the **$PERSISTENT** tree are displayed in a table with dynamic rows (*see screenshot below*).

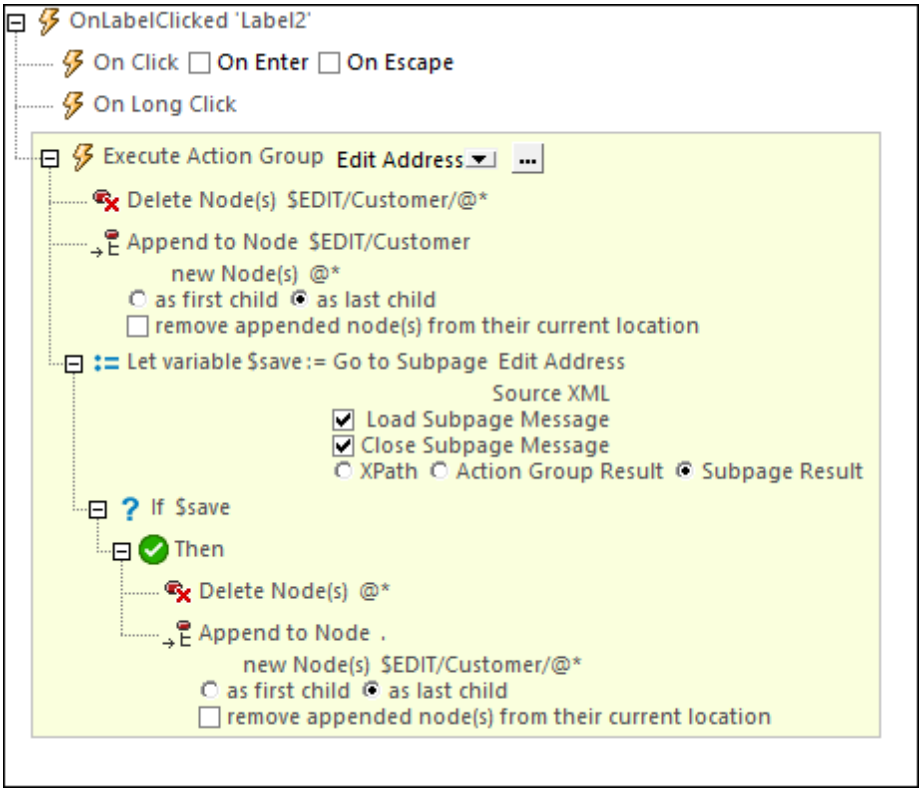
The table is defined as follows:

- 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** data 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 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*). Since the set of actions is the same for all four labels, the actions have been defined in a single Action Group that is reused on all four each labels.

Action Group to edit addresses

The Action Group 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 has been added is the **$PERSISTENT/Root/Customer** node. So this is the context node of all XPath expressions in the Action Group.

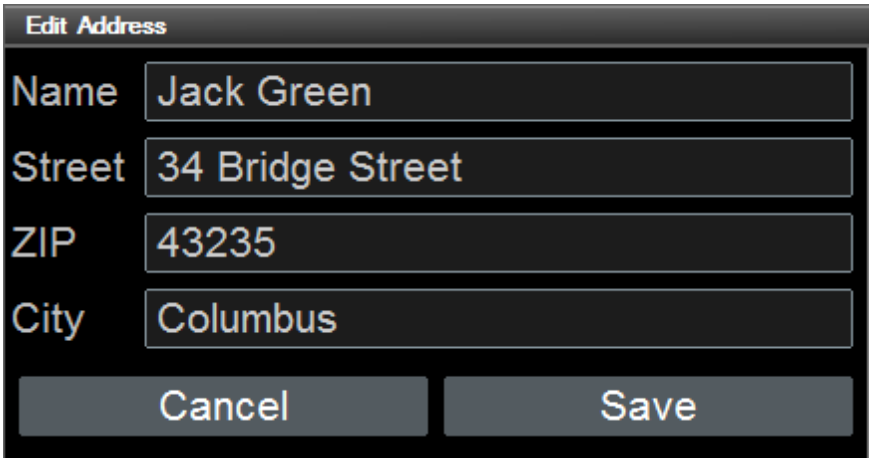


The actions in this action group do the following:

- The Delete Node(s) action deletes all the child attribute nodes of **$EDIT/Customer**. Remember that this data source contains the one customer record that is currently being edited.
- The deleted attribute nodes of the **$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: **$PERSISTENT/Root/Customer**. This replacement is done via the Append Node(s) action.
- The Let action creates a variable called **$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 and returns its result—which is the **Customer** node. So this node is then stored in the **$save** variable.
- An If-Then action next checks whether the **$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) 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 is executed without returning a result. The effect is to return to the main page without modifying the customer database in any way.

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.



- 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. 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.

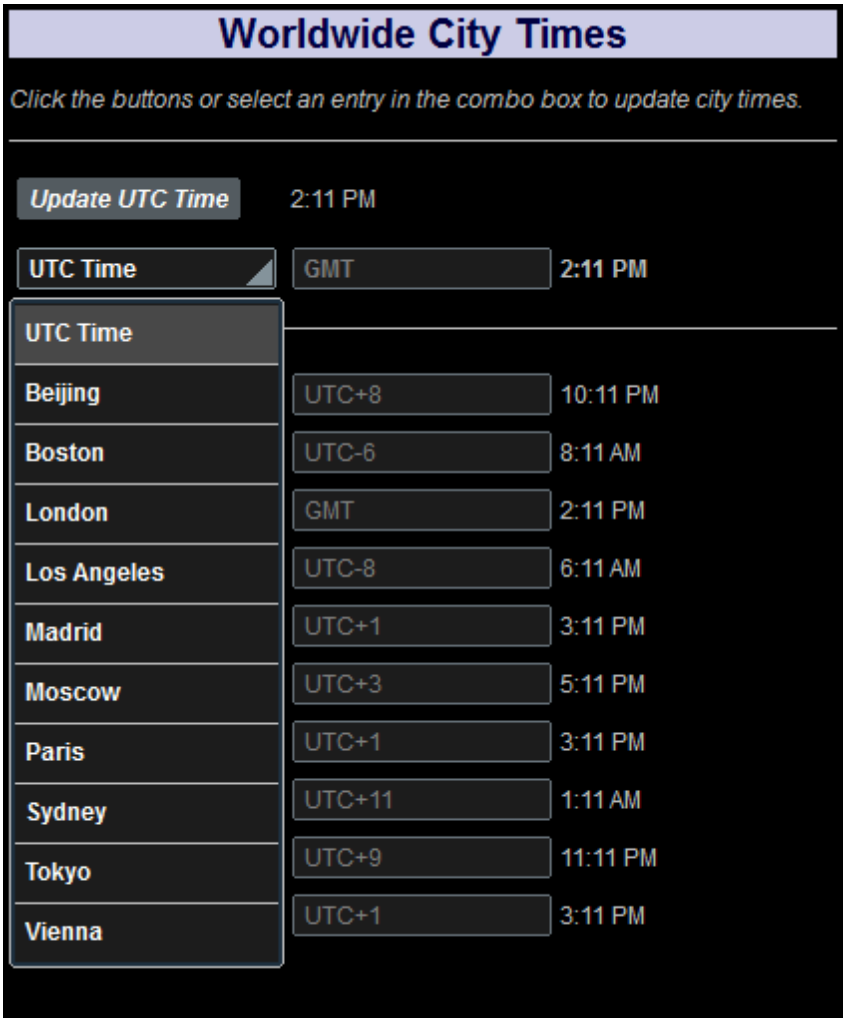
For the details of how the modified data replaces the old customer data in the customer database, see the description of the Action Group that carries out the relevant actions.

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 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.



The tutorial files

The files for this tutorial are located in your *(My) Documents* 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 describes the XML data source that is used for the structure and data of the design.
- Design Components describes the various controls and actions of the design.
- Refreshing the Page shows how values in the display can be automatically updated by actions defined for a page refresh.

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*):

- The **UTC** element, which is updated with the UTC time via a SOAP request when the user presses a button (*see listing below*). This node is used to display the UTC time to the user.
- The **RefreshTime** element (*see listing below*), 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*). 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*). (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 data 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 data source, CityTimes.xml

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

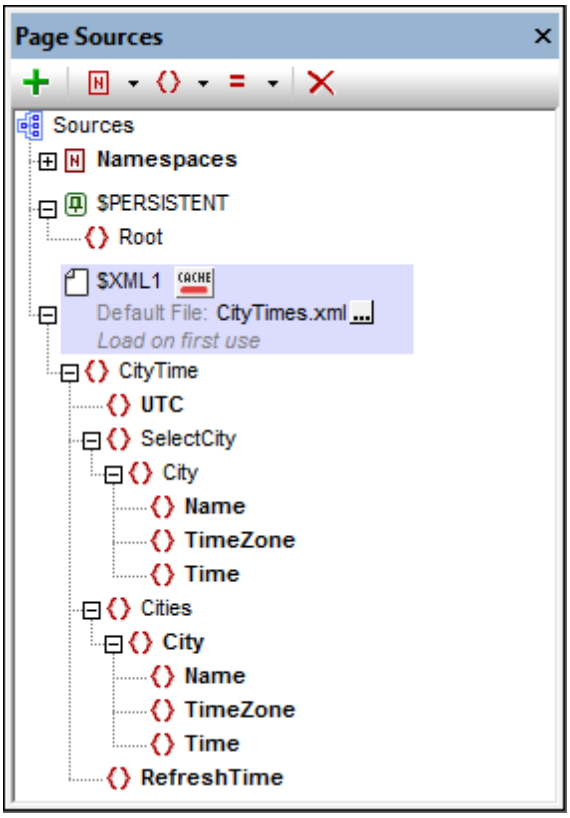
```
<?xml version="1.0" encoding="UTF-8"?>
<CityTime xmlns="http://www.Nanonull.com/TimeService/">
	<UTC>12:00 AM</UTC>
	<RefreshTime>60</RefreshTime>
	<SelectCity>
		<City>
			<Name>UTC Time</Name>
			<TimeZone>GMT</TimeZone>
			<Time>12:00 AM</Time>
		</City>
	</SelectCity>
	<Cities>
		<City>
			<Name>Beijing</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 data source has been added to the design's page sources. The page sources was added as follows:

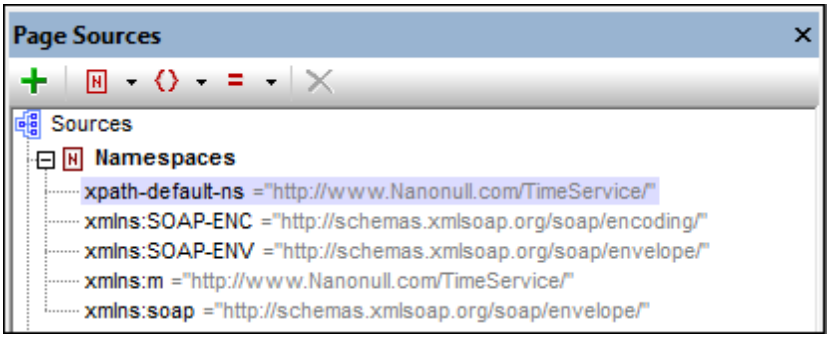
1. Click the **Add Page Source** icon in the Page Sources Pane (*see screenshot above*). Select *New XML, HTML or JSON structure imported from file*. Confirm the page source settings in the next screen (without changing the defaults) by clicking **Finish**.



2. An Open dialog appears. Browse for the **CityTimes.xml** file (*see location*), 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 (*screenshot below*).



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: .

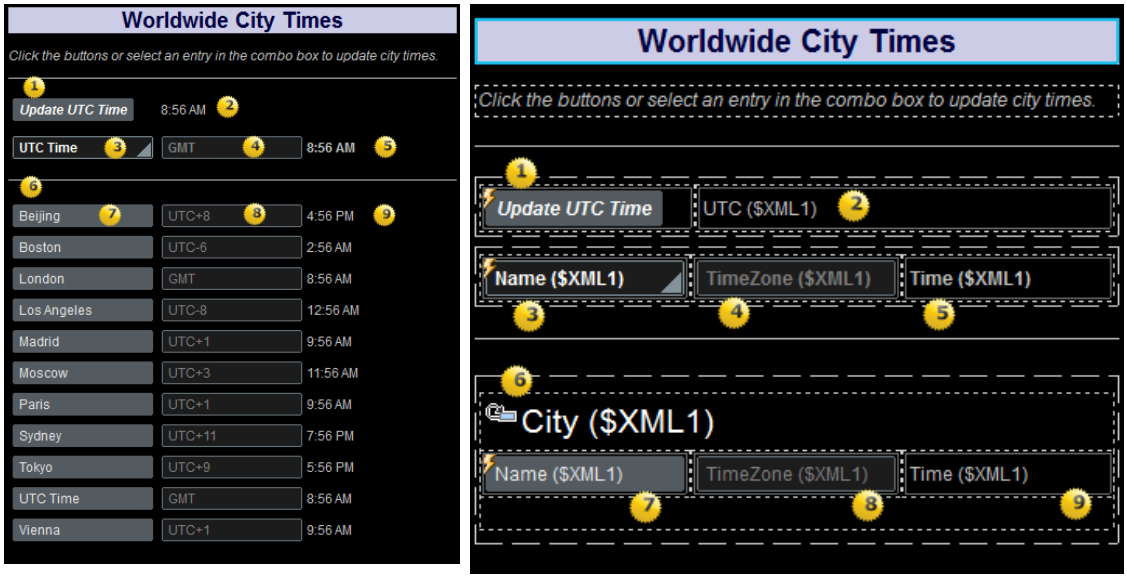
```
<CityTime xmlns="http://www.Nanonull.com/TimeService/">
```

- 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. 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.

Design Components

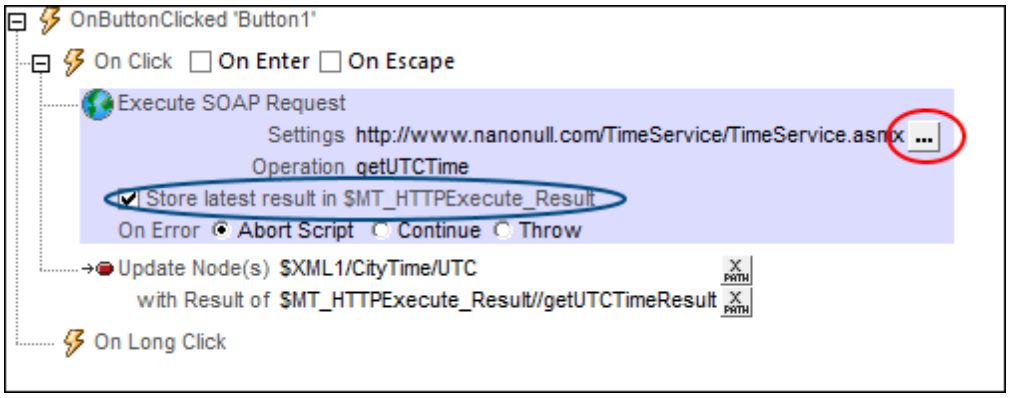
The design components are numbered in the screenshots below and are described in callouts further below. The screenshot at left shows the simulation, the screenshot at right shows the design. Click a callout to see a description of the corresponding design component.



Simulation of the solution at runtime. *The page design.*

▼ 1: Button to update UTC time

When the *Update UTC Time* button is clicked at runtime, the **`OnButtonClicked`** event triggers two actions (*see screenshot below*). First, an Execute SOAP Request action sends a SOAP request for the UTC time to the web service. The SOAP response from the web service is stored in the **`$MT_HTTPExecute_Result`** variable (*circled in blue in the screenshot below*). Next, an Update Node action updates the **`$XML1/CityTime/UTC`** node with the UTC time. The content of this node is immediately displayed in a label (*see Callout 2 below*).



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 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 **$XML1/CityTime/UTC** (the label's page source link). Data from this node will be displayed in the label. Since the node **$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

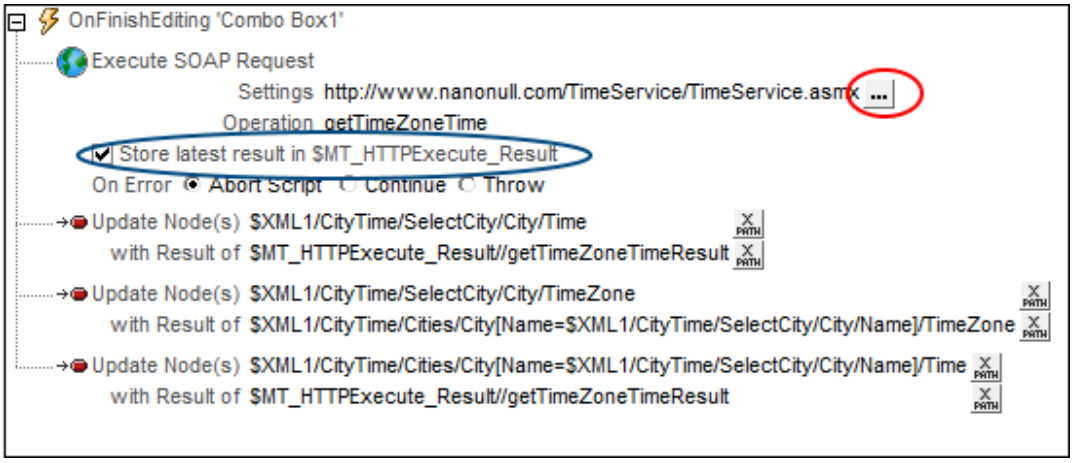
The purpose of the combo box is the following:

1. To display the names of the cities that are listed in the **Cities** element of the XML page source
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. The items of the combo box dropdown list are the names of the cities in the **Cities** element of the XML page source. 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 between the combo box and this XML tree node (and done by dragging-and-dropping the tree node from the Page Sources Pane 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 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 appears.



In the SOAP Request dialog, 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 **$i** variable. The expression then goes on to select (from the **Cities** element of the XML page source) the **Timezone** element of the city that has a **Name** element that matches the value in **$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** 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. 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*.

Updating nodes with the timezone time

The Update Node action is used to update two XML tree nodes with the received timezone

time: (i) **$XML1/CityTime/SelectCity/City/Time**, and (ii) **$XML1/CityTime/Cities/City[Name=$XML1/CityTime/SelectCity/City/Name]/Time**. The highlighted part in the second expression specifies that only that city in the **Cities** database should be updated that has a name that matches the name of the user-selected city. The content of these updated nodes are immediately displayed in labels via page source links (*see Callouts 5 and 9 below*). The value of the timezone time is obtained from the SOAP response via the **$MT_HTTPExecute_Result** variable.

Displaying the timezone of the selected city
The Update Node 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 (*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). 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). 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.

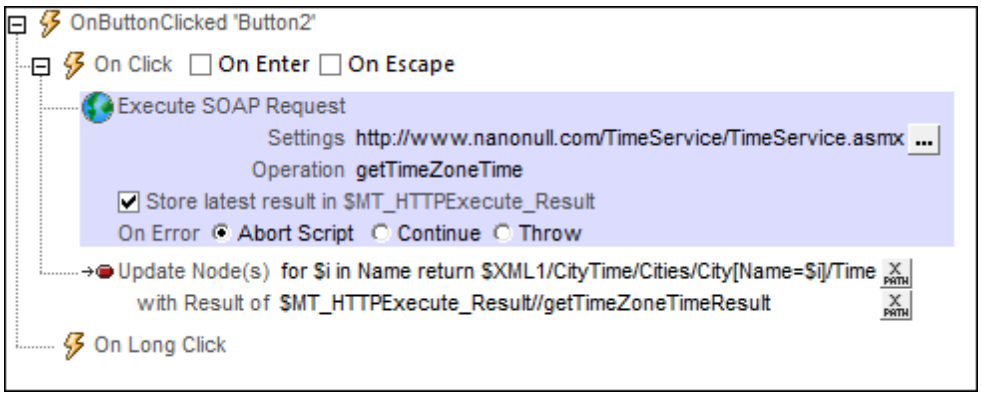
▼ 6: Table displaying the Cities database

The cities in the **Cities** element of the XML page source 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), 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** variable. Next, an Update Node 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*).



▼ 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). The content of this node does not change.

▼ 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). 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.

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:

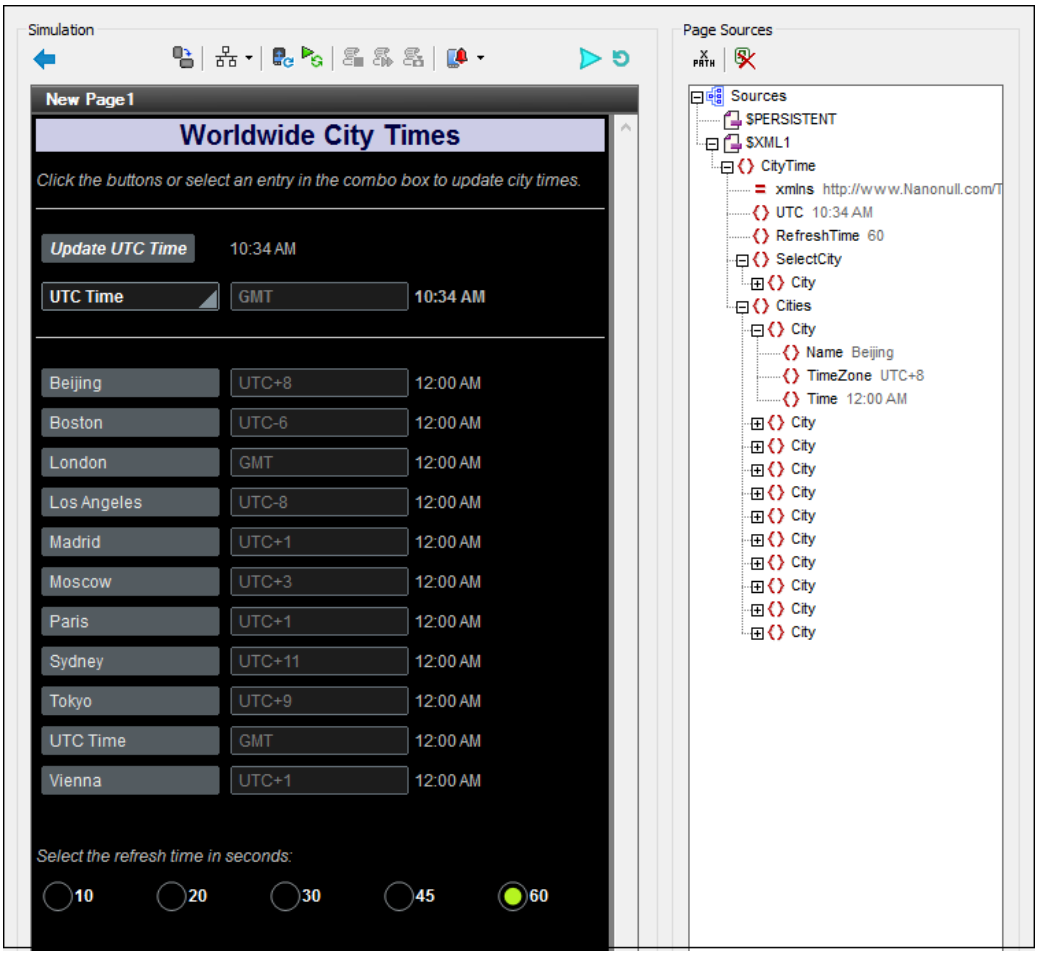
- Execute SOAP Request: The action requests the UTC time from the web service and stores the response in the **`$MT_HTTPExecute_Result`** variable. The request is defined in the same way as for the UTC Time button (*Callout 1*).
- Update Node: Updates the node **`$XML1/CityTime/UTC`** with the UTC time. Since this node is the page source link of the UTC Time label (*Callout 2*), the label will be initialized with the current UTC time.
- Update Node: 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*), we initialize the **`SelectCity//Time`** node with the current UTC time.

Refreshing the Page

The design components that we have described till now 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` event.

To automatically refresh the display of the city times, we have used the following mechanism.

- A node in the XML file has been defined to hold the amount of time, in seconds, between refreshes: **`$XML1/CityTime/RefreshTime`**
- The `OnPageRefresh` event has been defined such that the page refreshes: (i) when the user taps the device's **Refresh** button (*see screenshot below*), and (ii) after every **`X`** number of seconds, where **`X`** is the number in the **`$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.

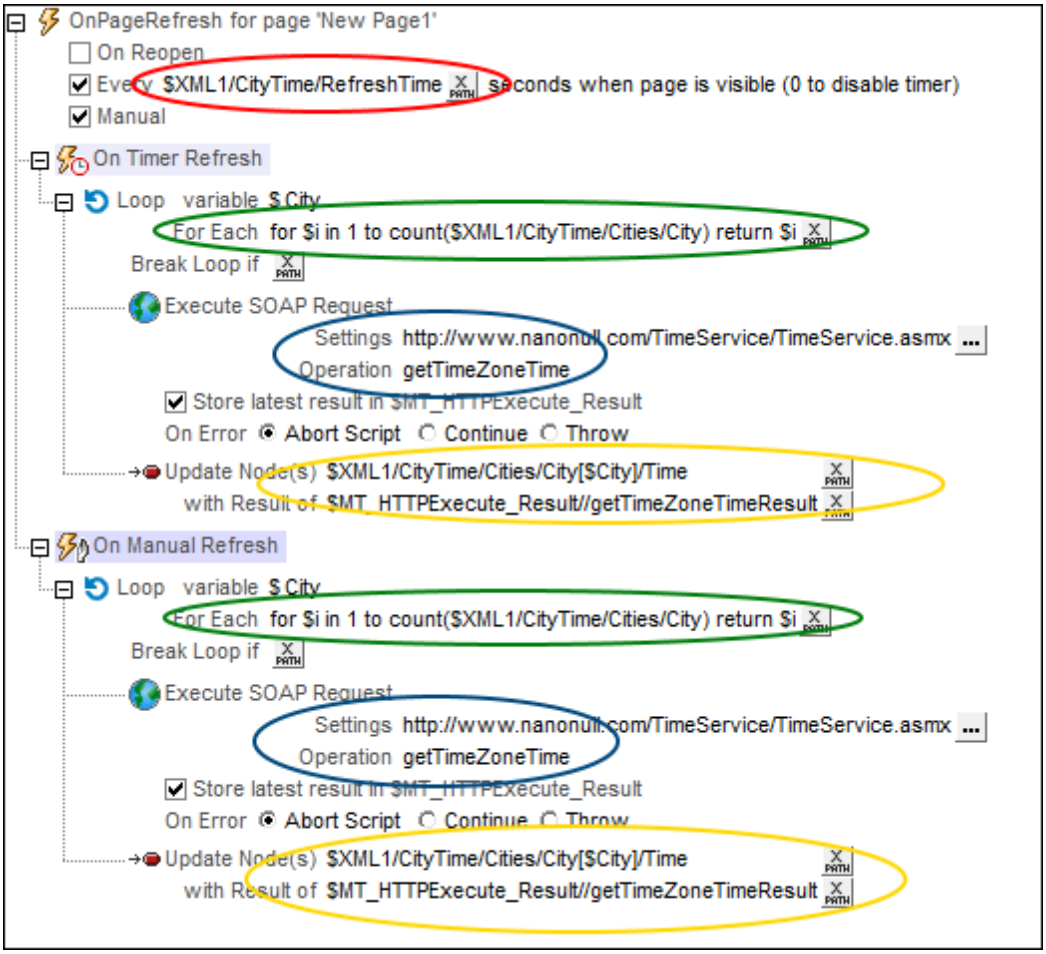


The key points of this mechanism are described below.

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*).



Since we wish to update the current time of each city in the display, we have done the following:

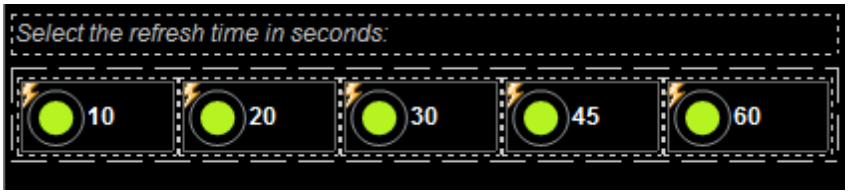
- Created a loop that iterates over each city (*definition circled in green*). The loop returns a sequence of integers. Each integer is tied to a **City** node by being the index of that **City** node. We do not wish to iterate directly over the City nodes because we wish to update these nodes within the loop, and updates are not possible while the nodes are being iterated over.
- 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 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 (*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 to the **$XML1/CityTime/RefreshTime** element. This means that they form a mutually exclusive set, and that the **Checked Value** of the selected radio button will become the content of the **RefreshTime** element.
- Each button has a Restart/Stop Page Timer action defined for its **OnFinishEditing** event. This is required in order to restart the page timer (defined in the **OnPageRefresh** event; *see above*) with the new refresh interval. Remember that the timer takes its refresh interval from the **$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. 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 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:

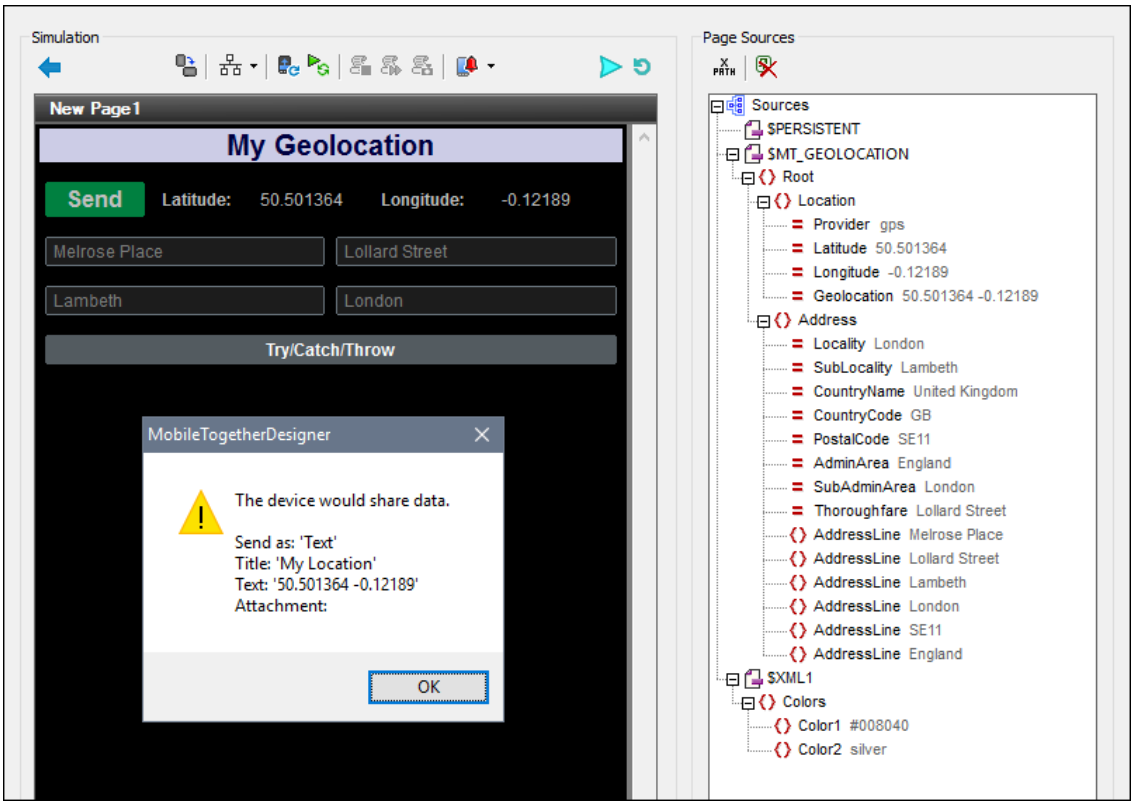
- Read the mobile device's current geolocation data and write this data into the design's **$MT_GEOLOCATION** tree
- Access the **$MT_GEOLOCATION** tree in order to display the geolocation data in the mobile device
- Share the geolocation data with contacts via the device's messaging and social networking 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.



The tutorial files

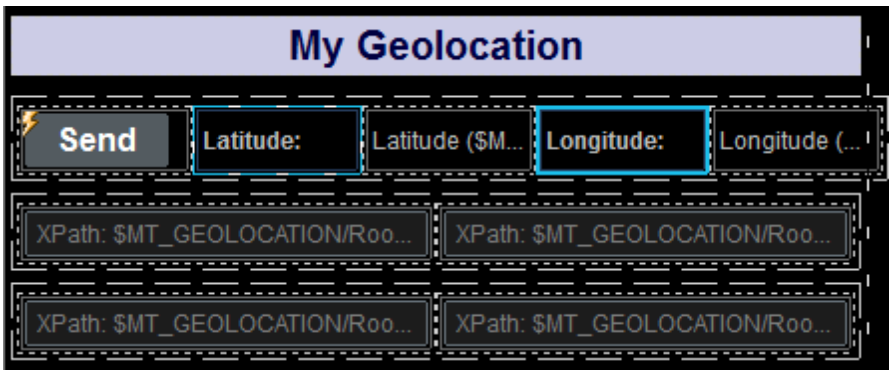
The files for this tutorial are located in your *(My) Documents* MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\Geolocations**.

- **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.

Reading and Sharing the Geolocation

The top part of the design (*screenshot below*) displays the geolocation data of the device.

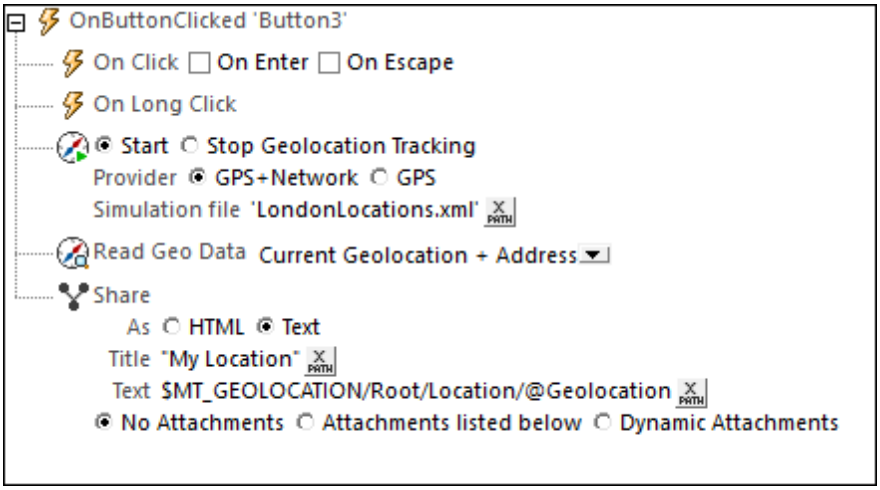


- The top row consists of the **Send** button and four labels. The two labels highlighted in blue use static text for their label text (`Latitude:` and `Longitude:`). The other two labels have page source links to the latitude and longitude nodes of the **`$MT_GEOLOCATION`** tree: **`$MT_GEOLOCATION/Root/Location/@Latitude`** and **`$MT_GEOLOCATION/Root/Location/@Longitude`**. As a result, whenever, these nodes are updated, the two labels will also be updated.
- The second and third rows contain a total of four Edit Fields. These controls are page source links, respectively, to the first four **`AddressLine`** nodes of the **`$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 to the data sources of the page when the Start/Stop Geo Tracking action or the Read Geo Data action is added to design.

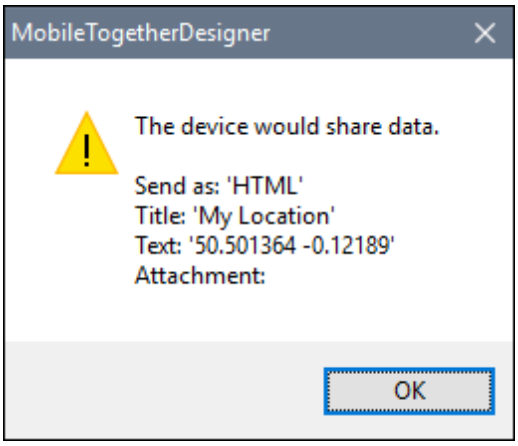
Send-button actions

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*).



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.

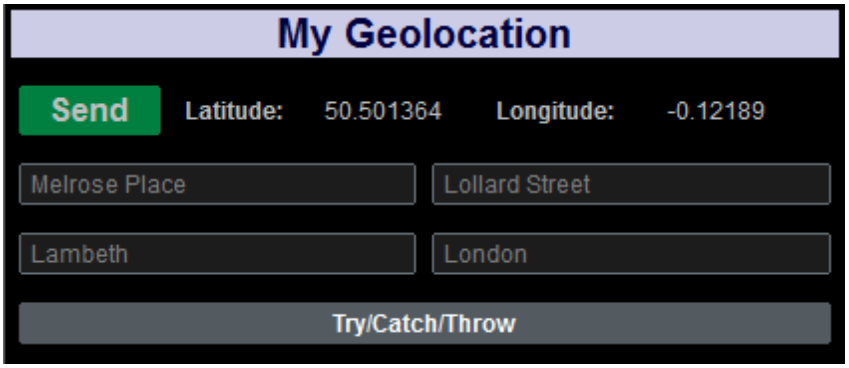
- Start Geolocation Tracking to start the tracking of the device. We have defined an XML file to be used as a simulation file. This file is called **LondonLocations.xml**, and it should be located in the same folder as the MTD file. Since actual geolocation data is not available during simulations on 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 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 **$MT_GEOLOCATION** tree is updated with the new geolocation data, all the labels and edit fields in the solution will automatically display the updated data. This is because of the page source links between the controls and the updated nodes.
- The Share action creates a text message with a title of *My Location*. the content of the message is the value of the **$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*).



For more detailed information, see the description of the respective actions.

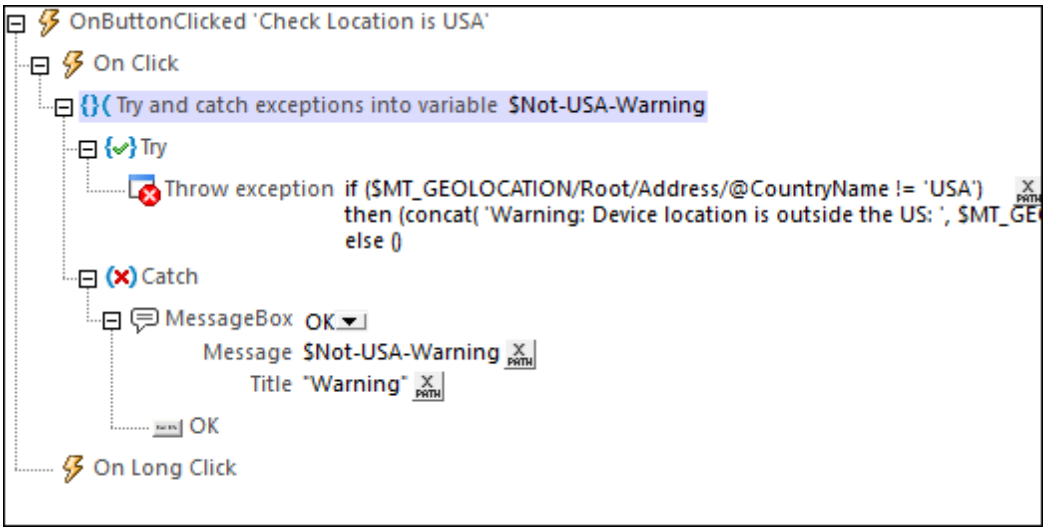
Using Try/Catch/Throw Exceptions

We have used the Try/Catch and Throw 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*).



The actions have been defined as follows:

1. A Try/Catch action was added.
2. We set up a variable **`$Not-USA-Warning`** which will be used to hold the exception message.
3. The Try part sets up a condition to test whether the geolocation is not in the USA. This condition is specified in the XPath expression of a Throw action. If the condition is true, then an exception is thrown. The exception message will be stored in the **`$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

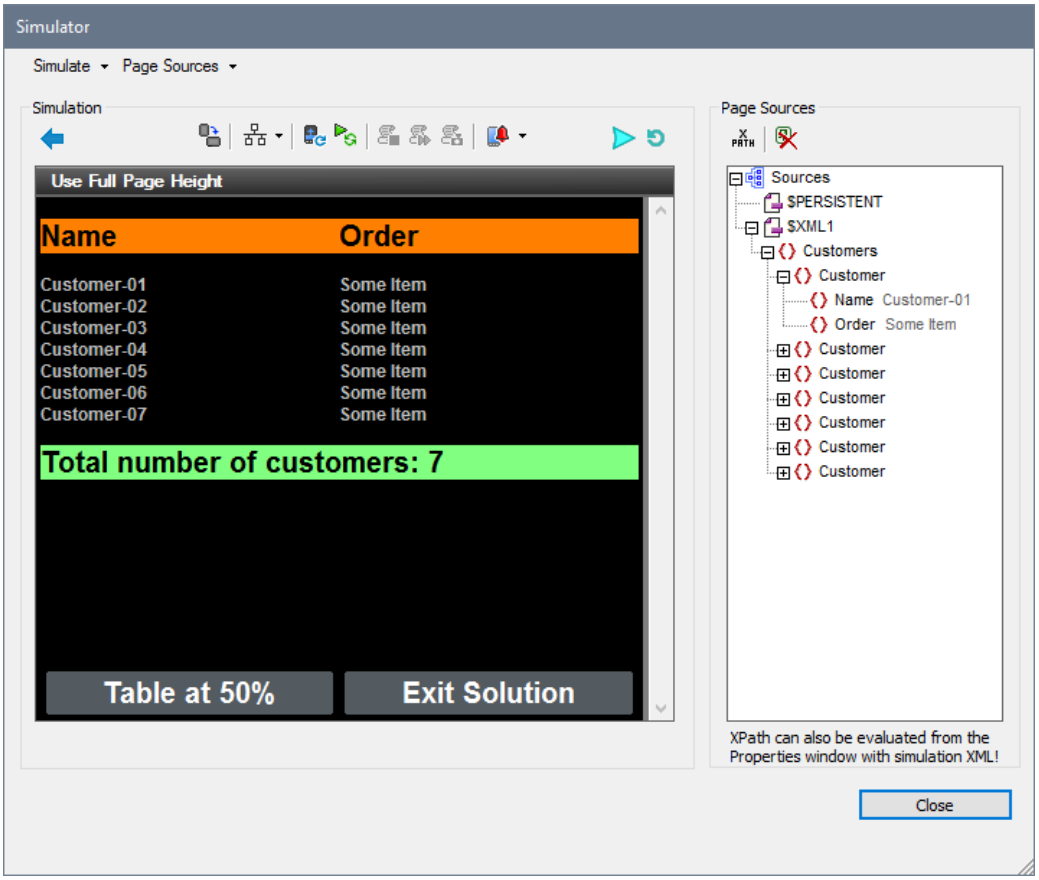
If the geolocation country **is not** USA, then the condition is `true` and the expression evaluates to the string generated by the **then** clause. Since this result is not an empty sequence, an exception is thrown and the generated string is stored in the Try/Catch variable `$Not-USA-Warning`.

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 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



The tutorial files

The files for this tutorial are located in your *(My) Documents* 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

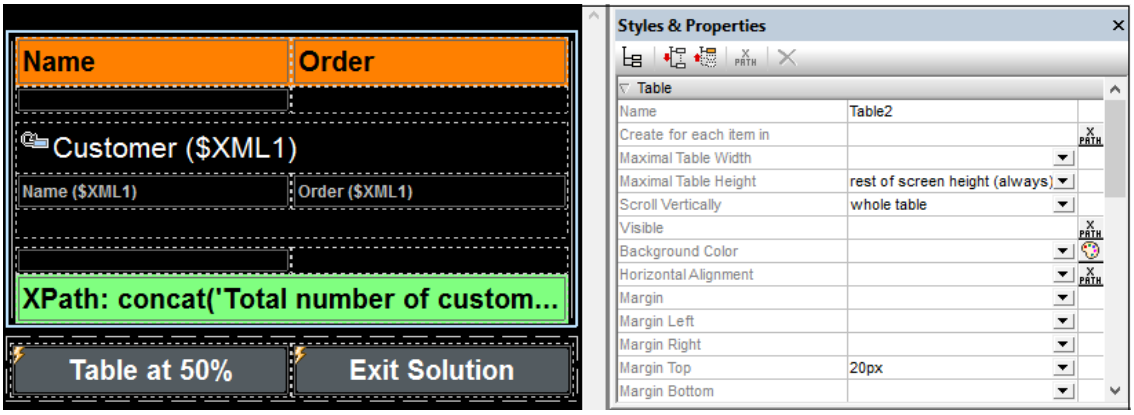
This tutorial is organized into the following sections:

- Tables that Force Full Screen Height describes settings to auto-adjust table heights so that the contents of the page fill the screen completely.
- Tables Having Specific Heights 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.

Tables that Force Full Screen Height

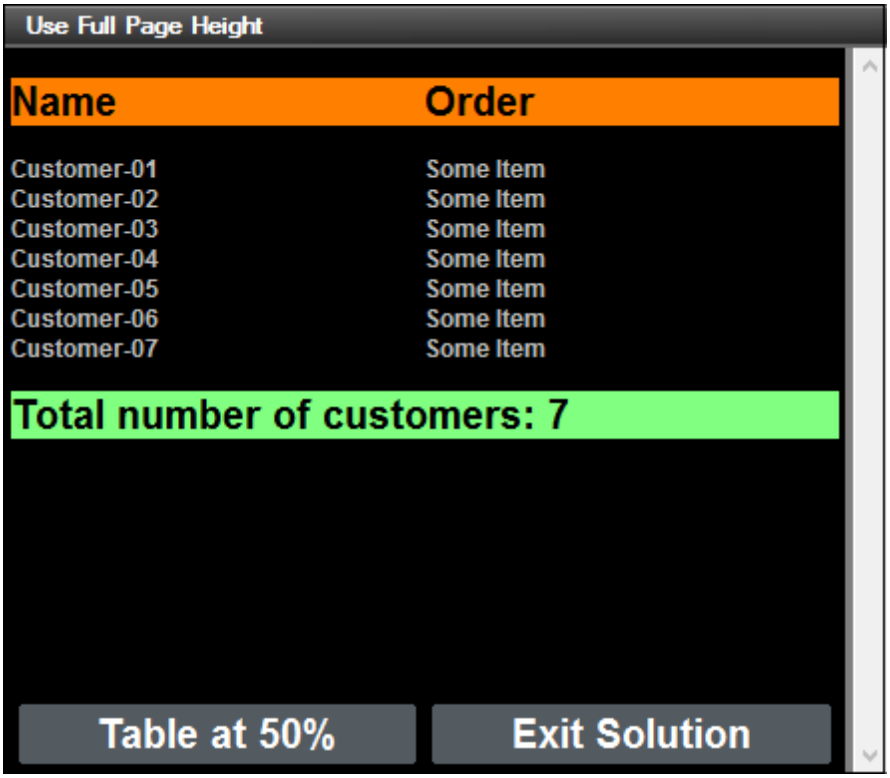
The first page of the design, *Use Full Page Height*, contains a table with dynamic rows that is created with the **`$XML1/Customers/Customer`** element as its repeating element (*see screenshot below*). This means that each **`Customer`** element is created as a Table Row Group and is displayed in its 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.



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 `Maximal 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.

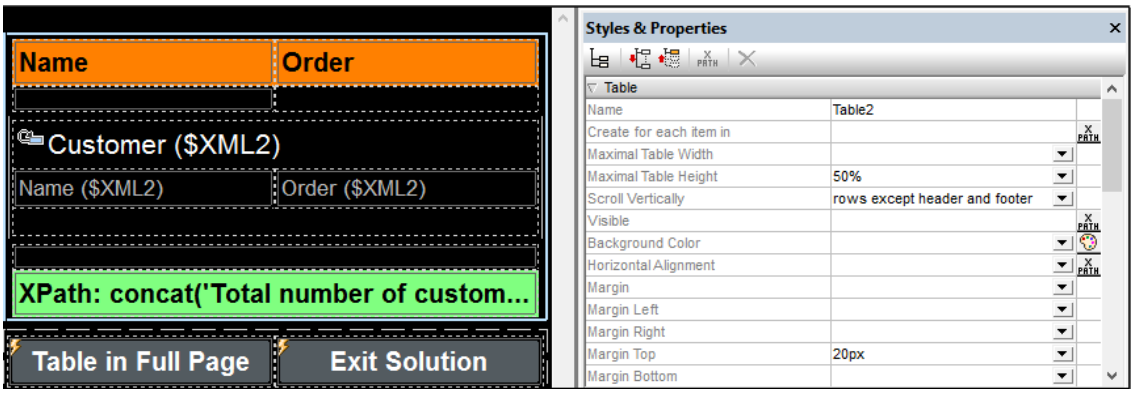


The point to note is the following: If the table's `Maximal Table Height` property is set to *Rest of screen height (always)*, then the height of the table will auto-adjust so that the components of the page are displayed to occupy the full screen.

You can modify the values of properties to test the various possibilities. See the section Table Properties for details of scrollable table properties.

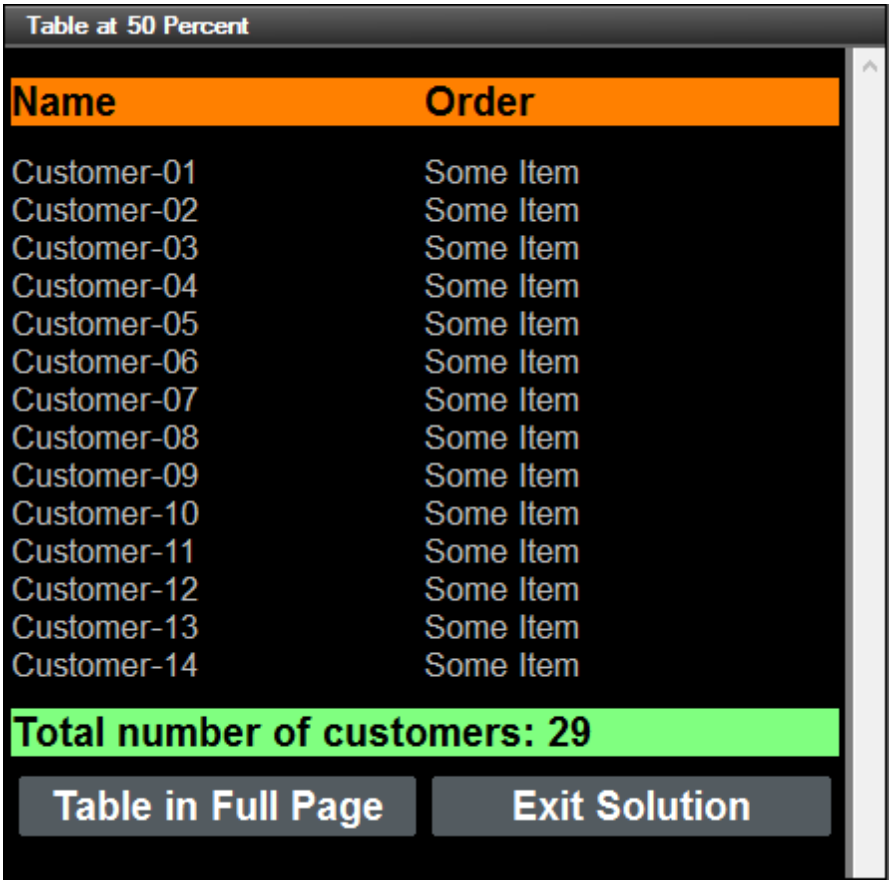
Tables Having Specific Heights

The second page of the design, *Table at 50 Percent*, contains a table with dynamic rows 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 Maximal Table Height property in the screenshot below*).



As a result of the *50%* `Maximal 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.



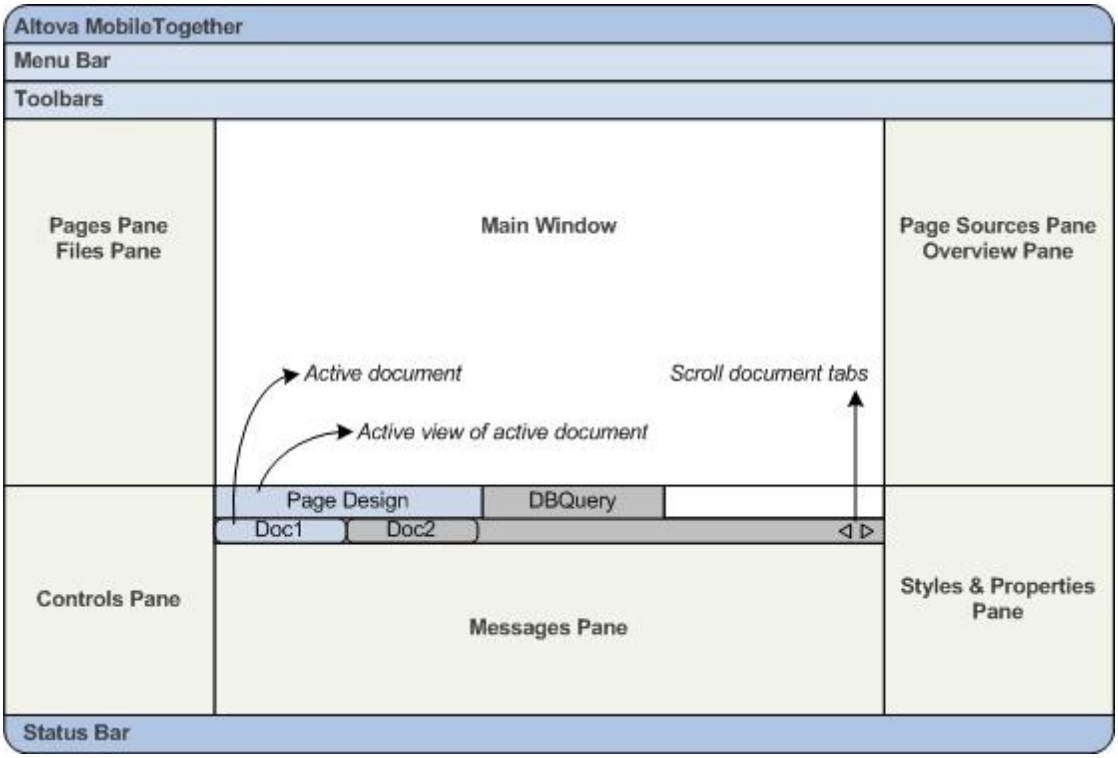
You can modify the values of properties to test the various possibilities. See the section Table Properties for details of scrollable table properties.

Chapter 5

User Interface

5 User Interface

The Graphical User Interface (GUI) consists of a Main Window (consisting of the Page Design and DB Query 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
- Files Pane
- Controls Pane
- Page Sources Pane
- Overview Pane
- Styles & Properties Pane
- Messages Pane

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.

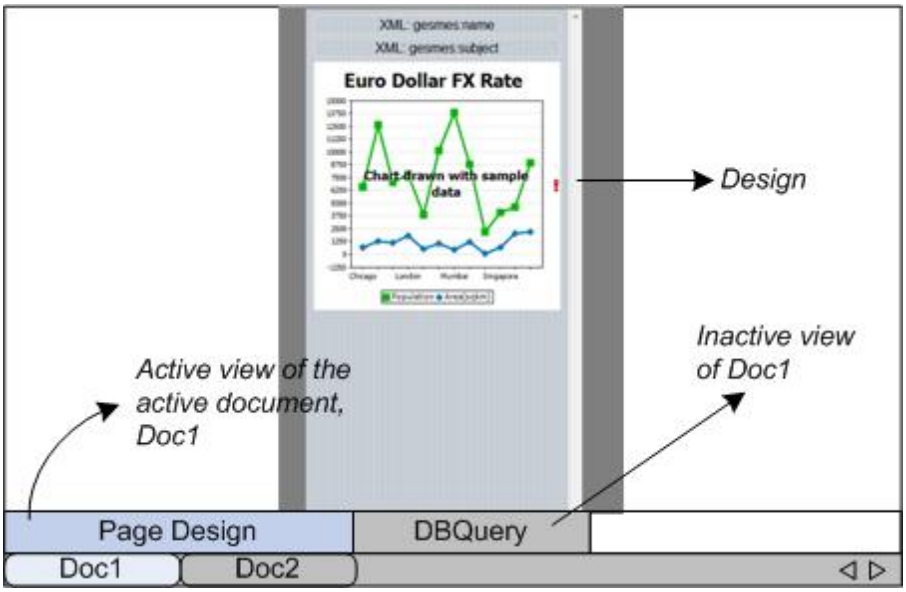
Auto-hiding panes

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 and DB Query.



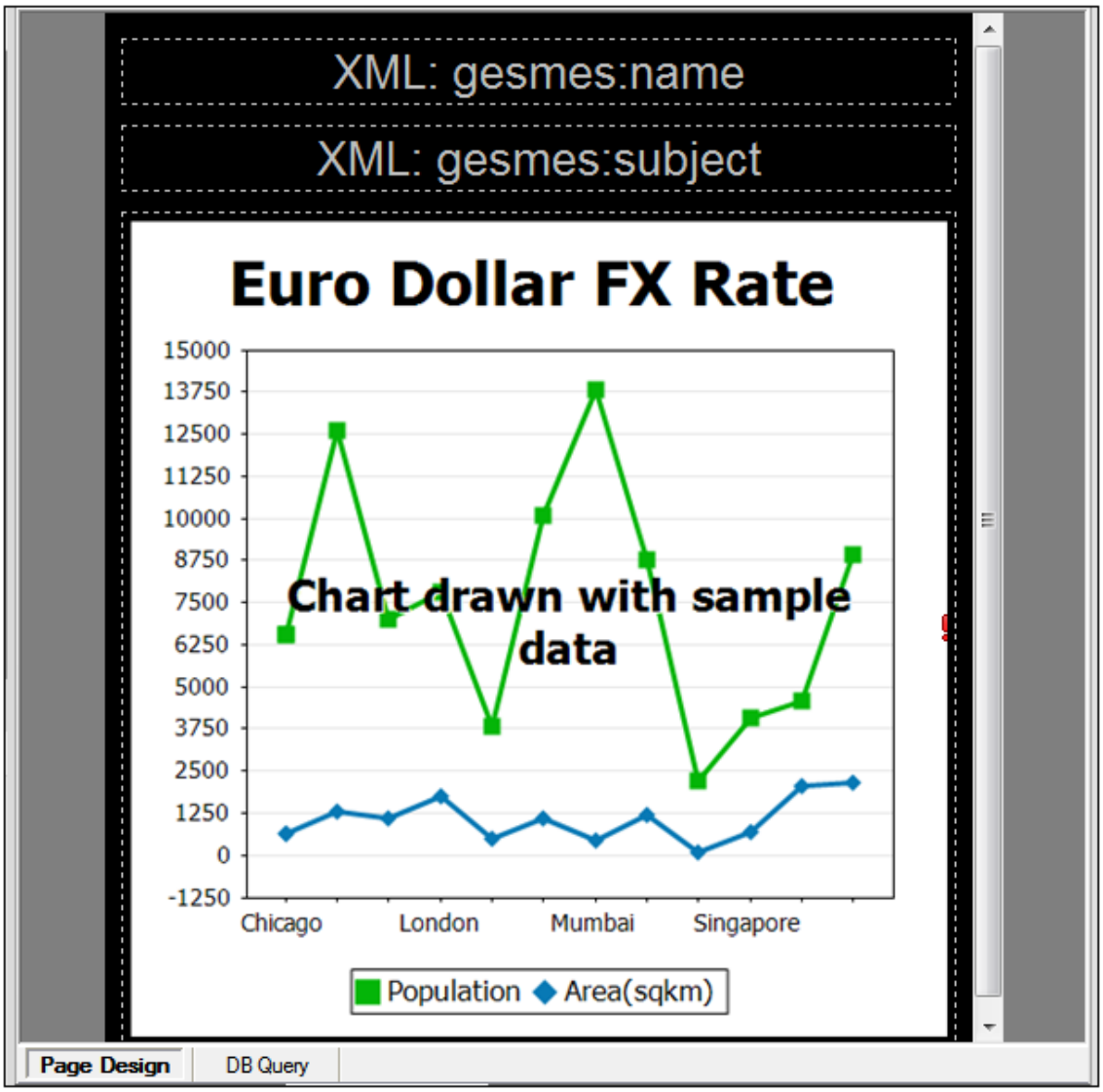
MobileTogether Design (.mtd) files in the Main Window

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** commands, such as **Print** and **Close**.
- Placing the mouse cursor over components in the Main Window displays a popup containing further information about the function of that component.

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 into the design, and then specify, in the Properties Pane, 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. Delete a control in the design by selecting it and pressing **Delete**.

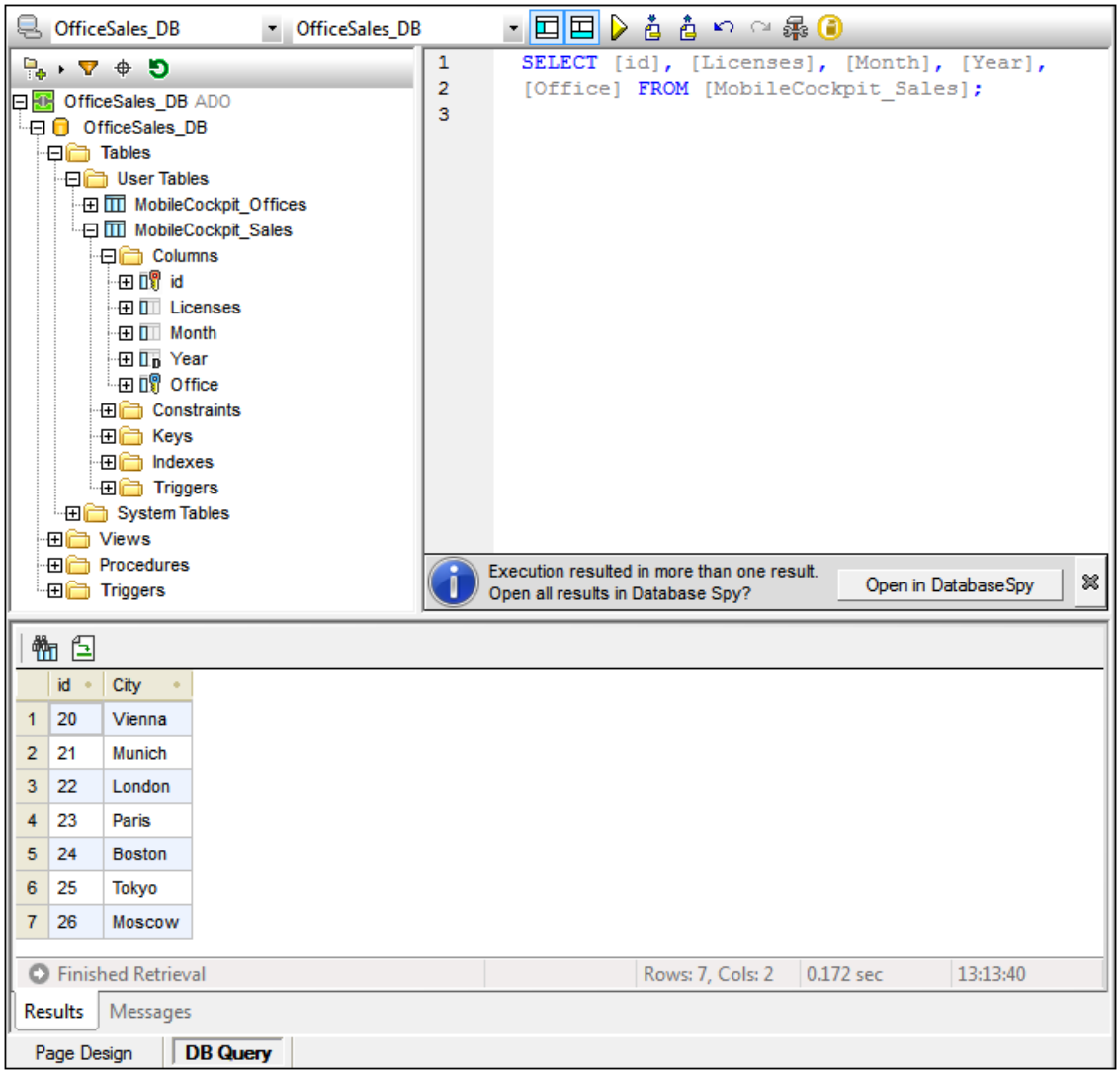
Page View settings

The screenshot below shows the Page View settings in the Main toolbar.

- *Device Selector:* This combo box allows you to select various mobile devices, so that the design can be previewed for specific devices.
- *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** menu.
- *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.

DB Query

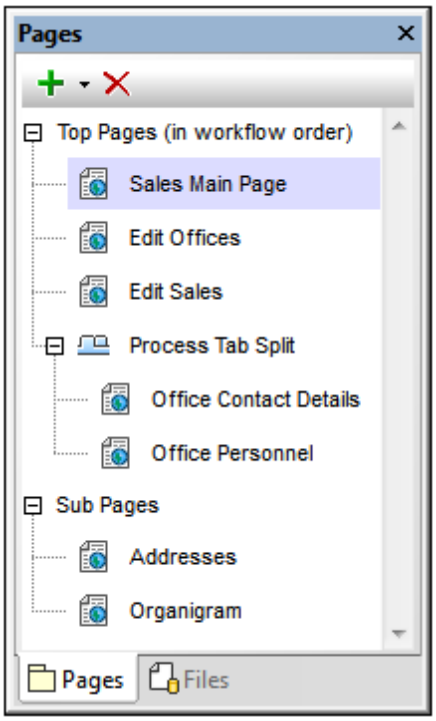
The **DB Query View** (*screenshot below*) enables you to directly query any major database from within the MobileTogether Designer GUI. The database could be a data source referenced in the active document or an external database. Note that each DB Query pane is associated with the currently active 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.



See the section, Database Query, for a detailed description.

5.2 Pages Pane

The **Pages Pane** 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).



To display a page, click its entry in the Pages Pane.

Top pages, sub pages, and tabbed pages

There are three kinds of project page:

- *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.
- *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).
- *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 (*Process Tab Split*) is defined to have two tabs that contain, respectively, the pages *Office Contact Details* and *Office Personnel*. Tabbed pages are part of the page sequence that makes up the project's workflow.

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**.

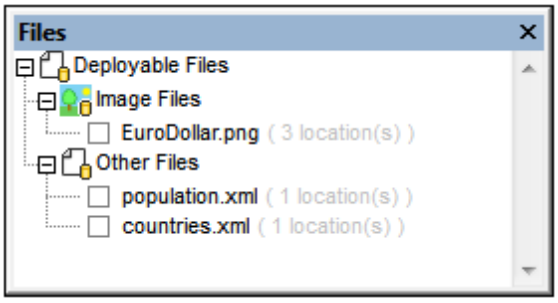
Context menu

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 after the last page of the *Top Pages* list	Right-click in either list and select **Add Tab Split**
Add a tabbed page before the selected page	Right-click and select **Insert Tab Split**
Find references in the design to the selected page	Right-click the page and select **List Usage in Actions**

5.3 Files Pane

The **Files Pane** (*screenshot below*) lists the files of the project that can be deployed to the server. Some files, such as the default files of data sources, are added automatically to the list of deployable files when these files are first associated with the project. But you can also add files directly to the list of deployable files via the context menu (obtained by right-clicking inside the Files Pane). These deployable files can be image files and other files, such as XML files, and they are organized in the pane under two headings: *Image Files* and *Other Files* (*see screenshot below*).



Each file has a check box next to it. To deploy a file, select its check box. To not deploy a file, de-select its check box.

The following information and actions are available in the Files Pane:

- Add or remove files via commands in the pane's context menu (*see below*).
- Make a filepath relative or absolute via the respective commands in the file's context menu.
- The total number of times the file is used across all the pages of the project is displayed next to the file name.
- Check the box in front of each file name to deploy the file to MobileTogether Server when the project is deployed to the server. Uncheck the box if you do not want to deploy the file.

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 and stored on the server, do not deploy it, but store it on the server at a location that the MTD file correctly references. (See the section Location of Project Files for details.) Files to be deployed are transferred to the server at the time when the solution is deployed. (See the section Deploying the Project for details.) When a file is deployed, it is stored with that design. 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.
- Do not deploy if the files needs to be written to.
- Do not deploy if the large size of files 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.

Adding and removing deployable files from within the Files Pane

To...	Do this...
Add a file	Right-click in the Files Pane and, from the context menu that appears, click **Add File**, and then browse for the file you want. The added file appears in the *Image Files* or *Other Files* list, according to what file type it is.
Remove a file	Right-click the file you want to remove and, from the context menu that appears, click **Remove File**.

Adding deployable files to the project from outside the Files Pane

Files can be deployed from outside the Files Pane is 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 data source in the Page Sources Pane.

Also see

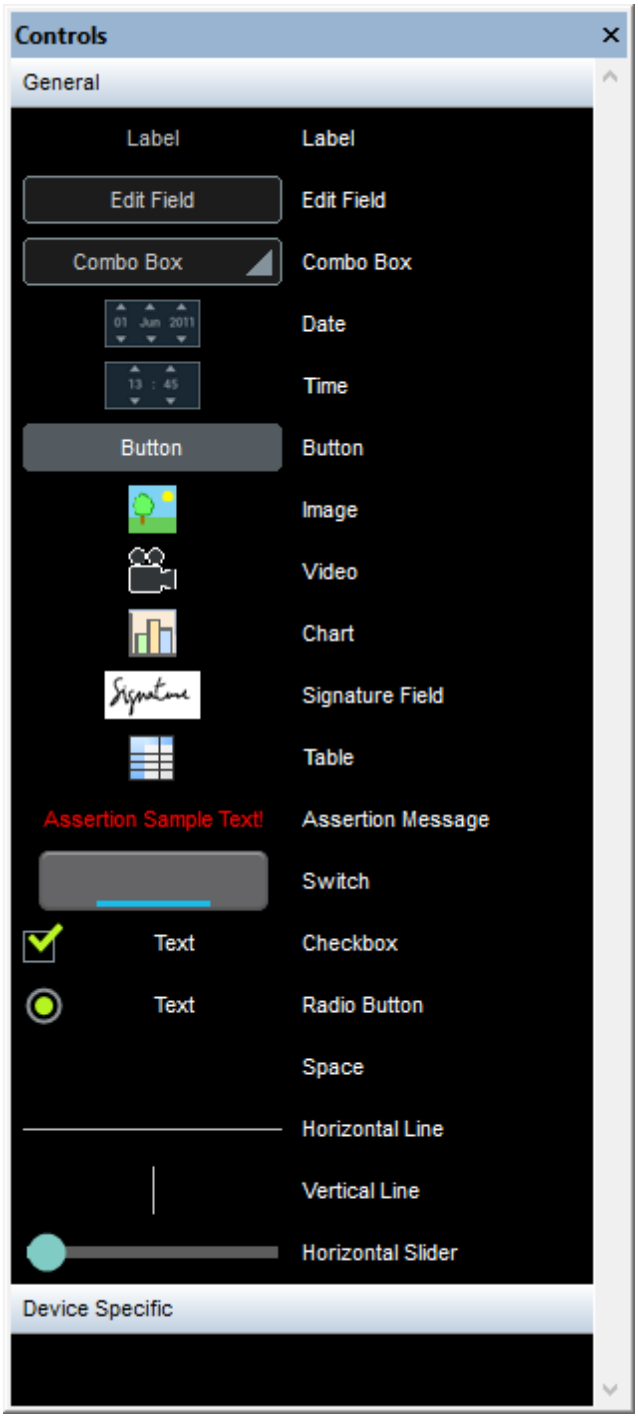
Deploy to MobileTogether Server
Location of Project Files
Deploying the Project
Data Storage on Servers.

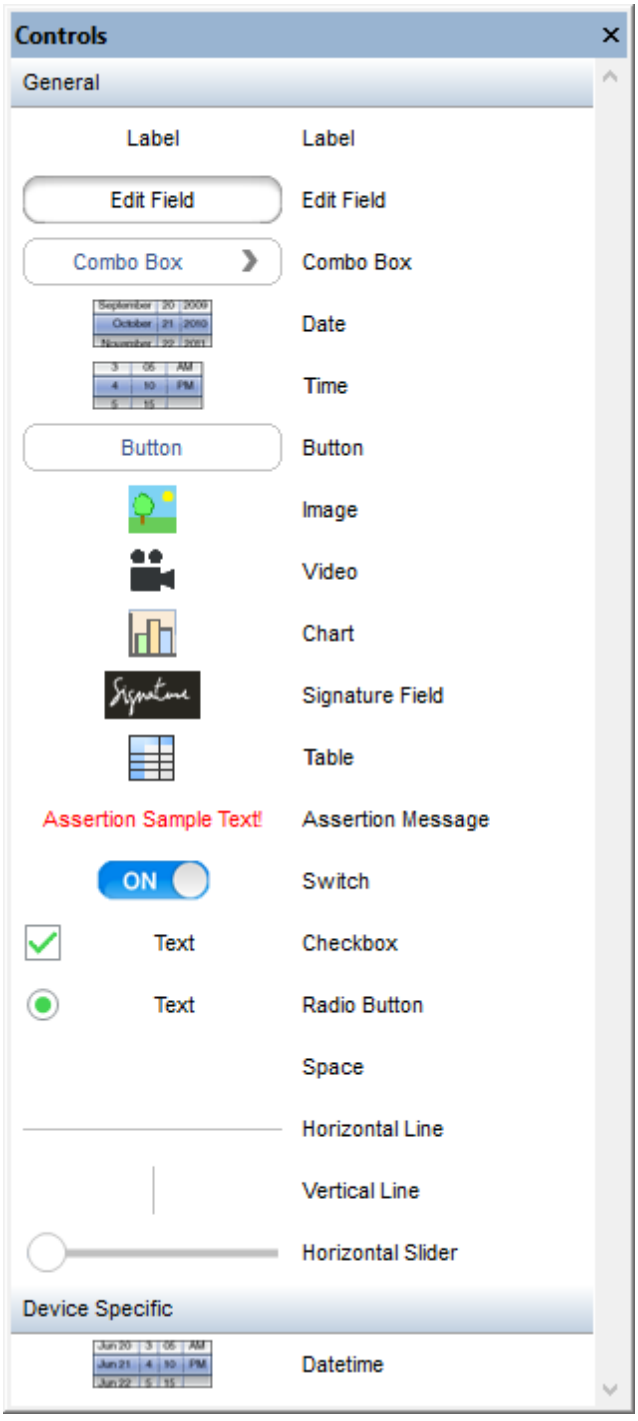
5.4 Controls Pane

The **Controls Pane** 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).

Page design controls

Page design controls are available when the Page Design tab is selected in the Main Window. The appearance of the Controls Pane and the controls corresponds to the currently selected device. For example, the Controls Panes shown in the screenshots below are for Android LG Optimus 7 (*left*) and iPhone 6 Plus (*right*).

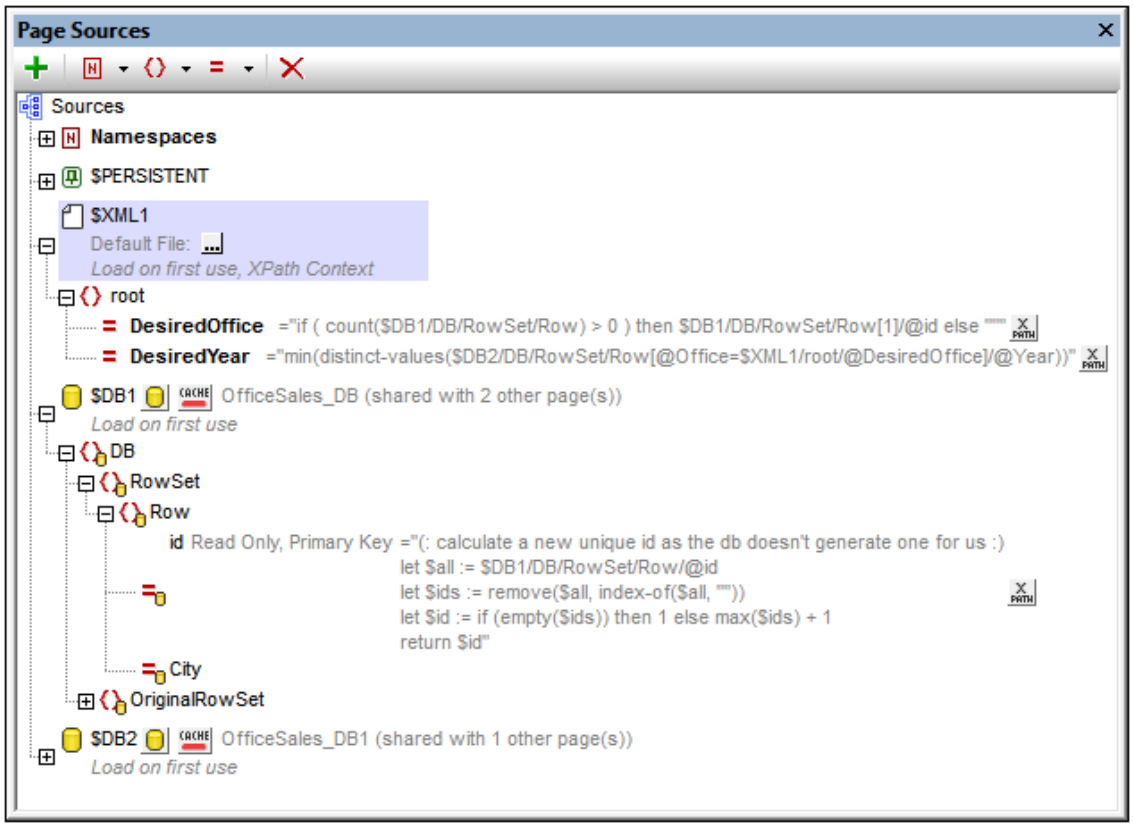

Controls
General
Label
Label
Edit Field
Edit Field
Combo Box
Combo Box
Date
Time
Button
Button
Image
Video
Chart
Signature Field
Table
Assertion Sample Text!
Assertion Message
Switch
Text
Checkbox
Text
Radio Button
Space
Horizontal Line
Vertical Line
Horizontal Slider
Device Specific



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.

5.5 Page Sources Pane

The **Page Sources Pane** (*screenshot below*) is where page data sources are managed. These data sources provide both (i) the structure of data trees used in the design, as well as (ii) the data held in the data trees.



The pane provides the following main functionality:

- Displays all the data sources of the page currently selected in the Pages Pane
- Displays all the namespaces declared for the active project
- Enables data 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 data source to be set. The data file provides the data that goes into the nodes of the data 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 data 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 Data Sources.

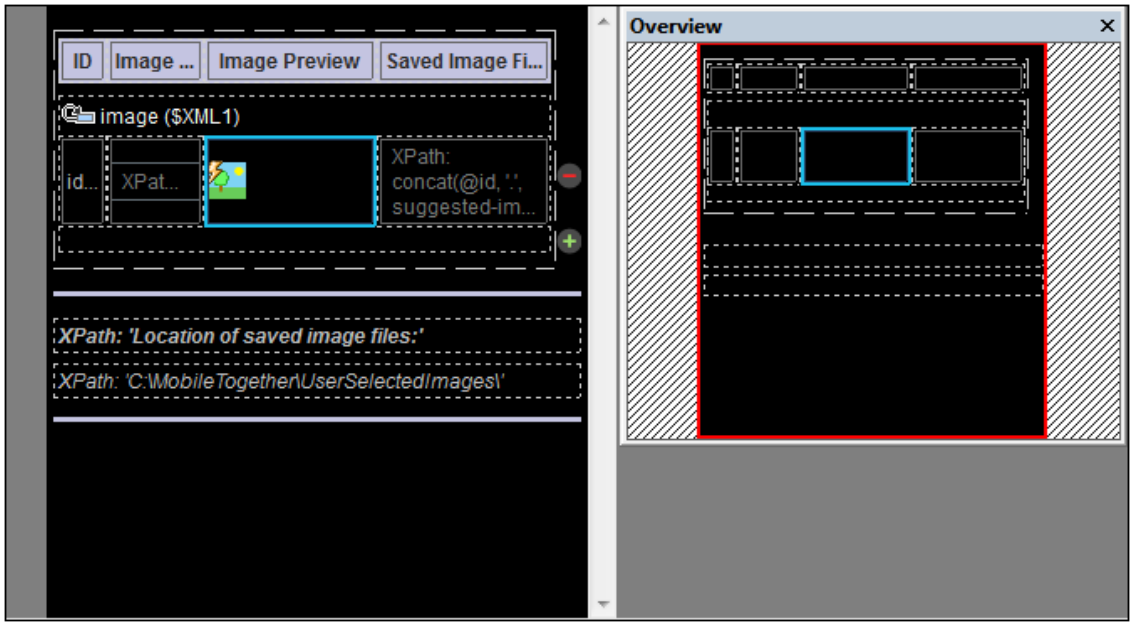
Manually creating a tree structure

Elements and attributes can be added relative to any node in a tree structure (including the root node), and they can be deleted. Select a node in a data 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.

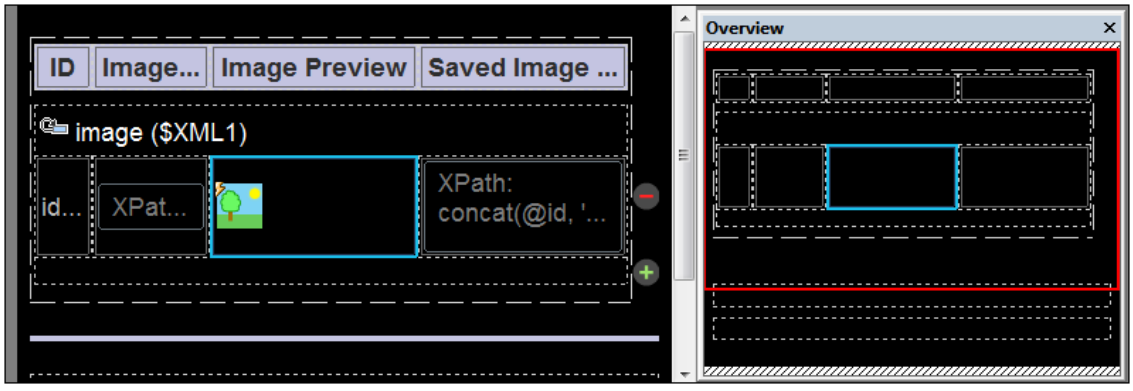
Icon	Command	Does this...
	**Add Source**	Displays the Add Page Source dialog. A root node is created for the data source that is added. Only one child element can be added to a root node.
	**Add Namespace**	Inserts or appends a namespace declaration under the *Namespace* entry. Edit the default prefix 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.

5.6 Overview Pane

The **Overview Pane** (*screenshot below*) shows a small version of the Page Design View. To see the default location of the Overview Pane, go to The Graphical User Interface (GUI).



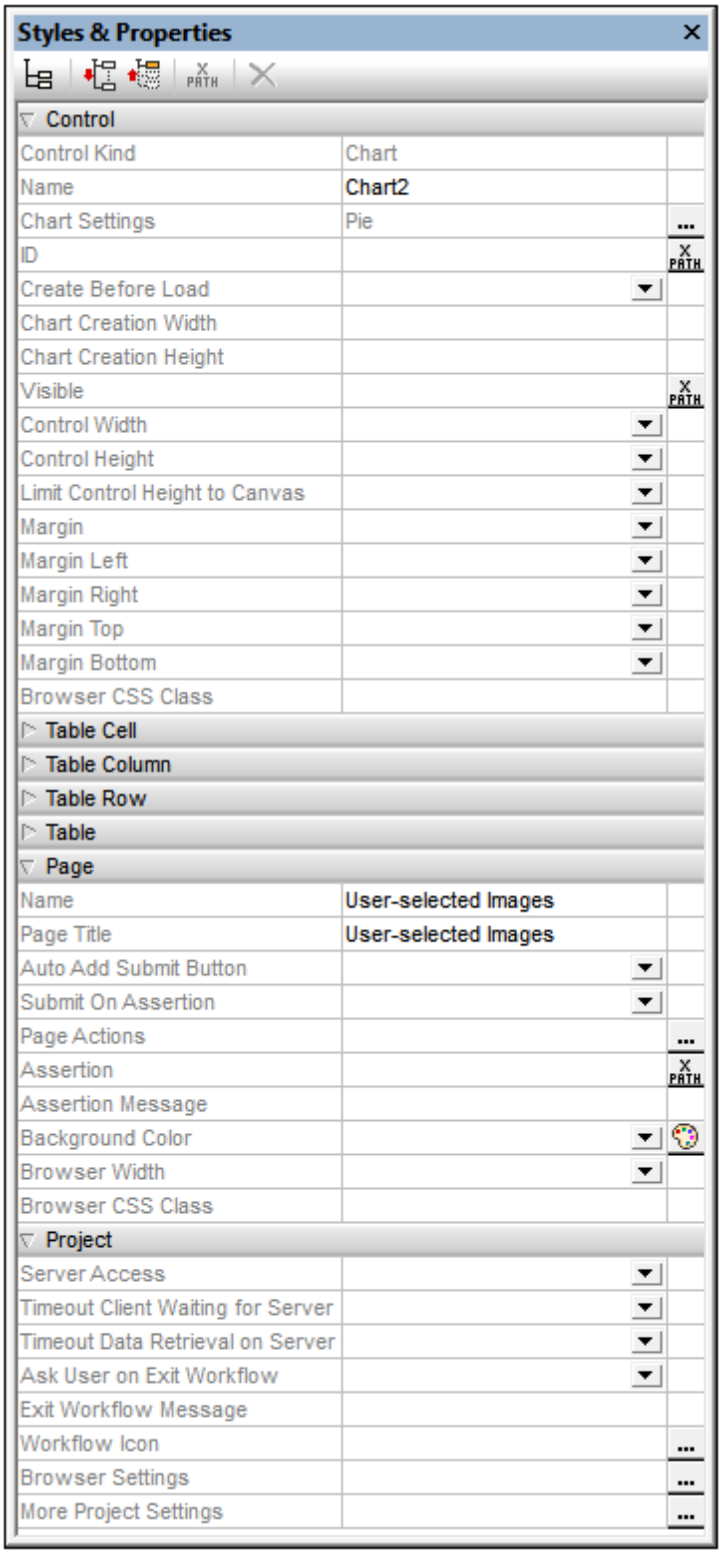
In the Overview Pane in the screenshot above, the extent of the design is indicated by the black area. The red rectangle is the viewport. It indicates the part of the design that is currently visible in Page Design View, and its dimensions correspond to those of the displayed part of the design (*compare 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.

5.7 Styles & Properties Pane

The **Styles & Properties Pane** (*screenshot below*) 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
Chart2
Chart Settings
Pie
ID
Create Before Load
Chart Creation Width
Chart Creation Height
Visible
Control Width
Control Height
Limit Control Height to Canvas
Margin
Margin Left
Margin Right
Margin Top
Margin Bottom
Browser CSS Class
Table Cell
Table Column
Table Row
Table
Page
Name
User-selected Images
Page Title
User-selected Images
Auto Add Submit Button
Submit On Assertion
Page Actions
Assertion
Assertion Message
Background Color
Browser Width
Browser CSS Class
Project
Server Access
Timeout Client Waiting for Server
Timeout Data Retrieval on Server
Ask User on Exit Workflow
Exit Workflow Message
Workflow Icon
Browser Settings
More Project Settings

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.
	Edit XPath	Enabled when a property that requires an XPath expression is selected. It displays up the Edit XPath/XQuery Expression dialog.
	Reset	Resets the property to empty or to its default value.

Entering and editing property values

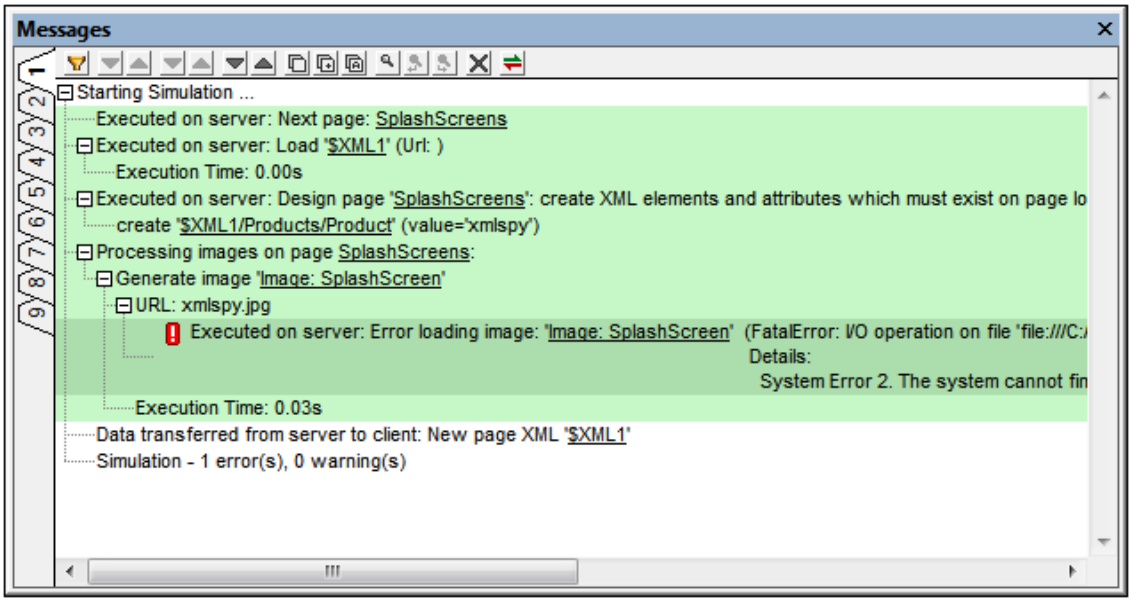
Property values are entered or edited depends on the type of entry field the property has:

Type of entry field	Do this...
Text field	Double-click, and enter or edit the property value.
Combo box	Select a value from the dropdown list.
Color palette	Pops up the color palette to enable selection of a color.
Edit XPath	Pops up the Edit XPath/XQuery Expression dialog to enable the entry of an XPath/XQuery expression. The expression provides the value of the property.
Additional dialog	Click to open the associated dialog. This could be, for example, the Actions dialog or the Open File dialog.

5.8 Messages Pane

The **Messages Pane** (*screenshot below*) displays messages in the following contexts:

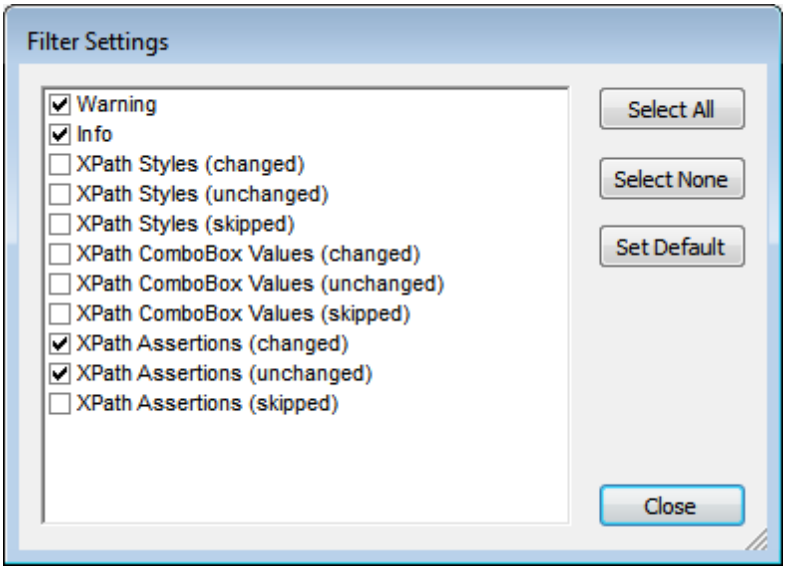
- It reports the validation results of the currently active project. This includes all pages in the project. If an error is reported, the error message contains a link that points to the component that generated the error.
- During simulations, it provides a detailed and step-by-step report of the progress of the workflow.



The toolbar of the Messages Pane contains commands that enable the following actions: filtering of the messages, navigation of the messages, copying of messages, searching the messages, setting the background colors of server and client log messages, and clearing the current tab. 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.

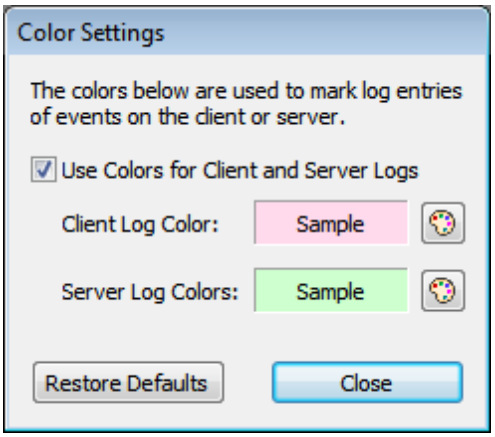
Filtering messages

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.



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.



Chapter 6

Project

6 Project

This section describes options that are relevant at the project level:

- Client-Server Interaction
- Location of Project Files
- Deploying the Project
- Project Properties
- Localization
- Namespaces
- Global Resources
- Performance

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.

A project's Server Access setting

A project's *Server Access* 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.

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.

6.2 Location of Project Files

The project file (aka design or MTD file), which has the **.mtd** extension, is deployed to the server. It is the solution that will be accessed by the MobileTogether Client app. When the project file is deployed to the server it is stored in a MobileTogether Server database, 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. When the file is deployed, it is the string that makes up the filepath in the design (relative or absolute) that will be used as the internal file reference on the server database.
  - For the details of deployment, see: Deploying the Project, Files Pane, Deploy to MobileTogether Server.

- *Embedded within the project file*
  - This applies to XML data files (typically the default files of data 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 in the Page Sources Pane, and select **Embed XML in Design File**. You can select whether files are automatically re-embedded 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, 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 data source, security authorizations can be set when specifying the properties of the data 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
  Files Pane
  Deploying the Project
  Data Storage on Servers.

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.

Deployment and access control

The available deployment options provide you with considerable flexibility in controlling access to solutions. There are two broad levels of access control.

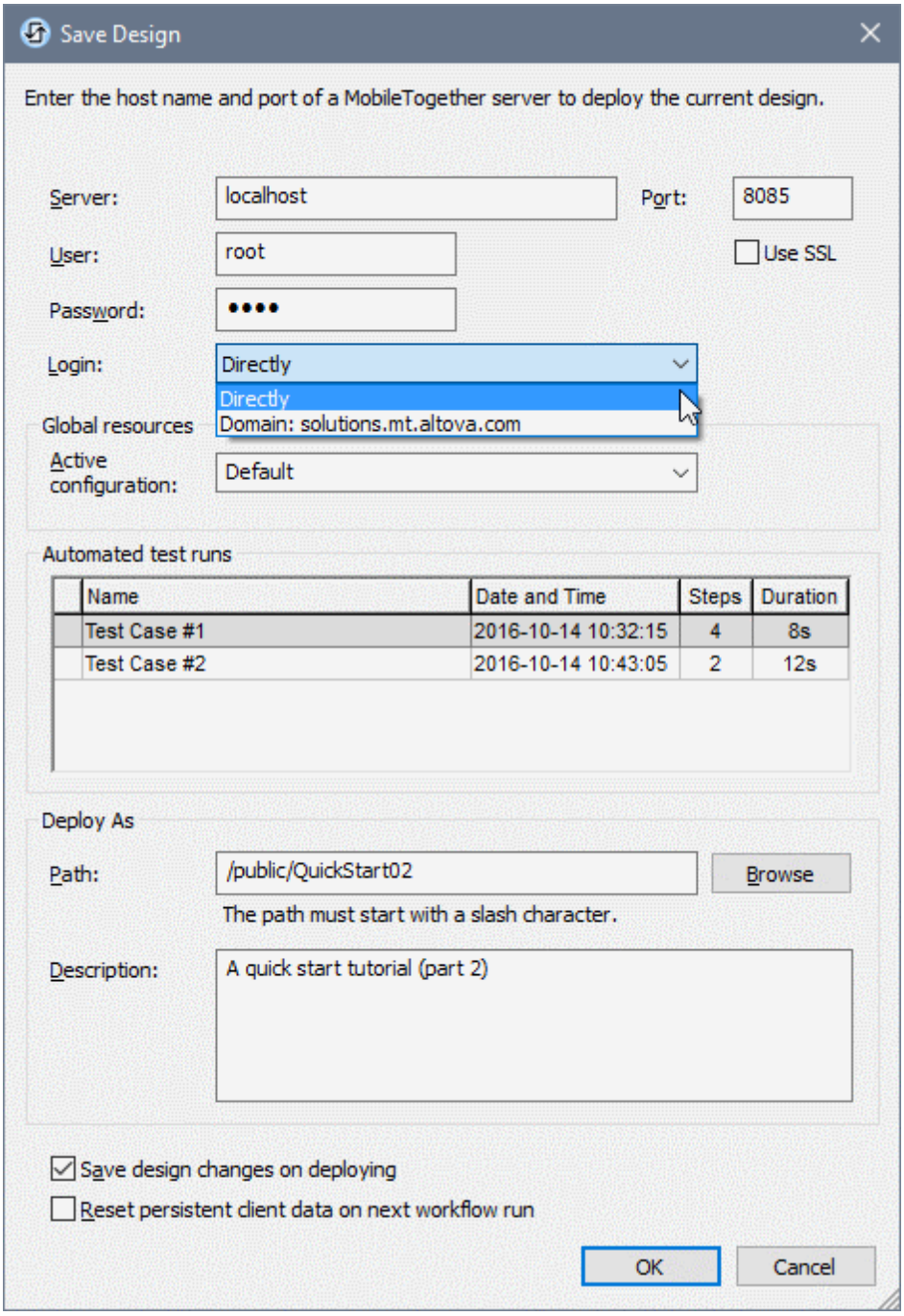
On a first level, access can be controlled at the server level, according to the kind of server access (internal/external) that is allowed:

- 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.

On a second level, for each server a set of users can be defined that have access to the solutions on that server. Access will be available only to those clients 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.

How to deploy a project

A project is deployed to the server with the **File | Deploy to Server** command. This command displays the Deploy to MobileTogether Server dialog (*screenshot below*), in which you specify the server connection details and whether the server uses SSL communication.



What is deployed?

The following files are deployed when the project is deployed with the **File | Deploy to Server** command:

- The project file (aka design or MTD file), which has the **.mtd** extension, is deployed to the server. This file is the solution that will be accessed by the MobileTogether Client app.
- All the deployable files in the Files Pane that have their check boxes selected. These files are typically image files and the default files of data sources.

All deployed files will be stored on the server and will automatically be correctly accessed by the solution file. This is very convenient because you do not have to worry about file paths being correct. Note, however, that these files are read-only. So, any files that need to be written to cannot be deployed, but must be stored manually on the server. The server and the design file must then be correctly configured to access the writable file. See the section Location of Project Files | A directory on the server for details of how to do this.

Note: Audio and video files **cannot** be deployed to MobileTogether Server via the MobileTogether Designer project's Deploy to Server mechanism. 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 action to load the binary audio/video data to a data source node; (ii) use the Save Binary action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 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.

Deployed files and the locations of project files

Deployed files are read-only. If there is a Save action defined for a file that is marked for deployment (in the Files Pane), then the design will be invalid—since the file will be read-only when deployed and cannot be written to. Deployed files are saved in the design and are read from there.

Files that are not deployed must be stored at a server location that is correctly referenced in the design. To build a correct reference to the file, you must correctly configure (i) the file's location when it is added, and (ii) MobileTogether Server's *Server Side Solution's Working Directory* setting. See the section Location of Project Files | A directory on the server for more information.

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** 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** function (described here). 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.

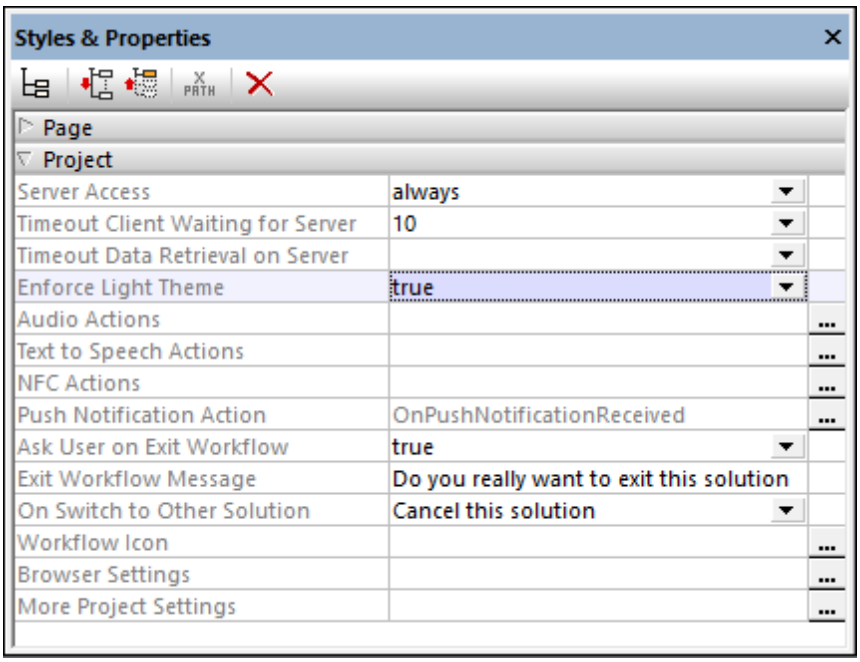
Note: Links to update server settings do not work in Gmail and some other email applications, but they work in popular clients such as AquaMail, K9, and MailWise. They have been tested in AquaMail and K9 and work correctly in these applications.

Also see

Deploy to MobileTogether Server, for a description of the Deploy to Server dialog (*screenshot above*)
Location of Project Files
Files Pane
Data Storage on Servers.

6.4 Project Properties

Project properties are defined in the Styles & Properties Pane and are described below. If you place your mouse over the property name, a popup appears that contains a brief description of the property.



▼ 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` tree, other persistent data, or embedded data. You can use the XPath function **`mt-has-serveraccess`** 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.

▼ Enforce Light Theme

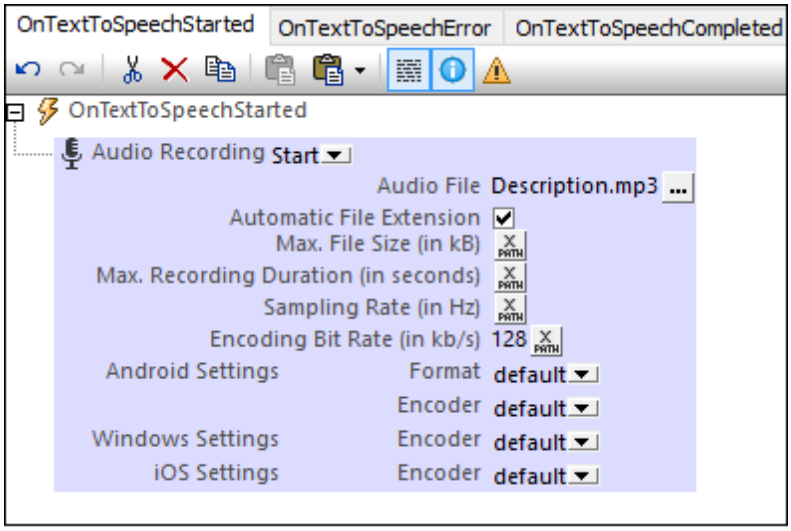
A light theme is one in which dark text is displayed on a light background. You can ensure a light theme for the entire project by setting the value of this property to `true`. If you set the value to `false` (the default value), then the current theme of the client device is used.

▼ 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.*

▼ 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*).



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 starts. For example, as shown in the screenshot above, an Audio Recording action 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:

- **`OnPushNdefMessageCompleted`** specifies what action/s to carry out when the transmission of NFC data (via NFC Push) has been completed.
- **`OnNfcTagDiscovered`** specifies what (additional) action/s to carry out when an NFC tag is discovered.

Click the property's **Additional Dialog** button to go to the definitions of the two events. See NFC-related events 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.

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* 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 for more information.

▼ 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** button on the first page of a solution. The user will **not** be asked for an exit-confirmation if the **Submit** button is pressed or if a Cancel Action Execution 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` 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 action, which is another way to specify whether a solution is canceled or minimized.

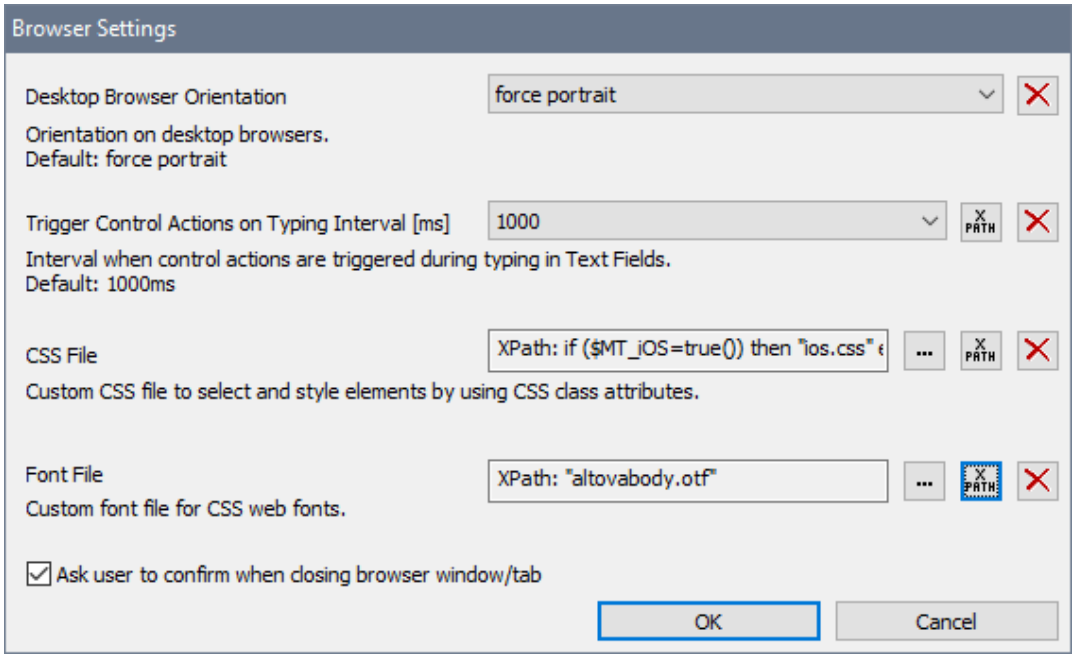
Note: Web clients do not support suspended solutions; only the active solution is supported.

▼ 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. 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.

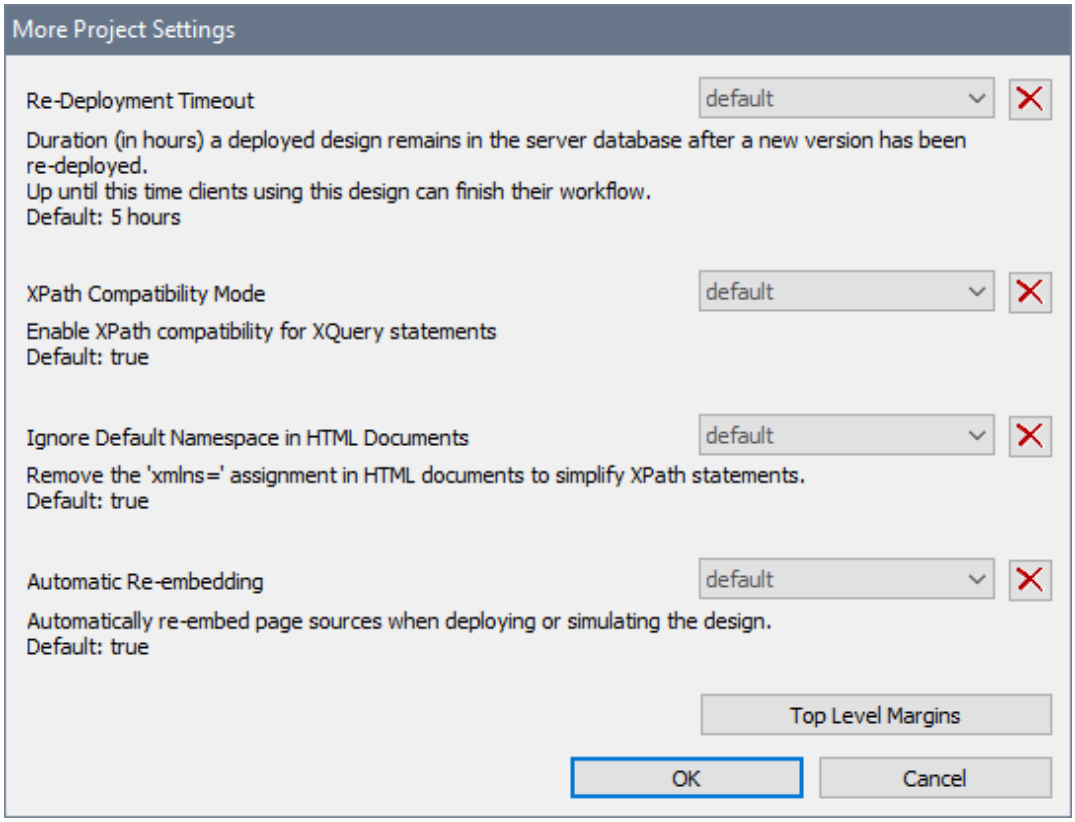


The following settings can be defined:

- *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*.
- *Trigger Control Actions on Typing Interval:* This setting applies to web clients only (as opposed to MobileTogether Client apps on mobile devices). Since updates of solution pages that are edited in web clients must be sent to the server for processing, it is useful to be able to specify when the updated data should be sent to the server. The value selected here is the time interval after which updated data is sent. The server processes the updated data and returns it so that all affected page components are refreshed. The default of this setting is `1000ms`. If the setting is disabled (by selecting `disabled` in the dropdown list of the combo box), then the control action is triggered when the end user changes focus, for example, by clicking somewhere else on the page. An XPath expression can be used to obtain the value of the setting, which must be either the string `'disabled'` or a number that is read by MobileTogether Designer as being the number of milliseconds.
- *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. You can also use an XPath expression to generate the file path. Alternative CSS files for different clients can be specified by using XPath's `if...then...else` conditional construct (*see screenshot above*). Note that the CSS rules defined in the external CSS file have a lower priority than the definitions that are made in a control's properties.
- *Font File:* Specifies a font file 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. A font that is embedded in the design in this way can be referenced via the `font-family` property of CSS.
- *Ask user to confirm when closing browser window/tab:* Displays a message box asking whether the user wants to leave the page. The message also informs the user that if he or she clicks Leave,then unsaved changes might not be saved. The user can additionally choose whether to prevent the page from creating additional dialogs or not.

▼ More Project Settings

Clicking the *More Project Settings* property's **Additional Dialog** button displays the More Project Settings dialog (*screenshot below*). You can select whether files are automatically re-embedded when the user starts a simulation or deploys the solution to the server.



The following settings can be defined:

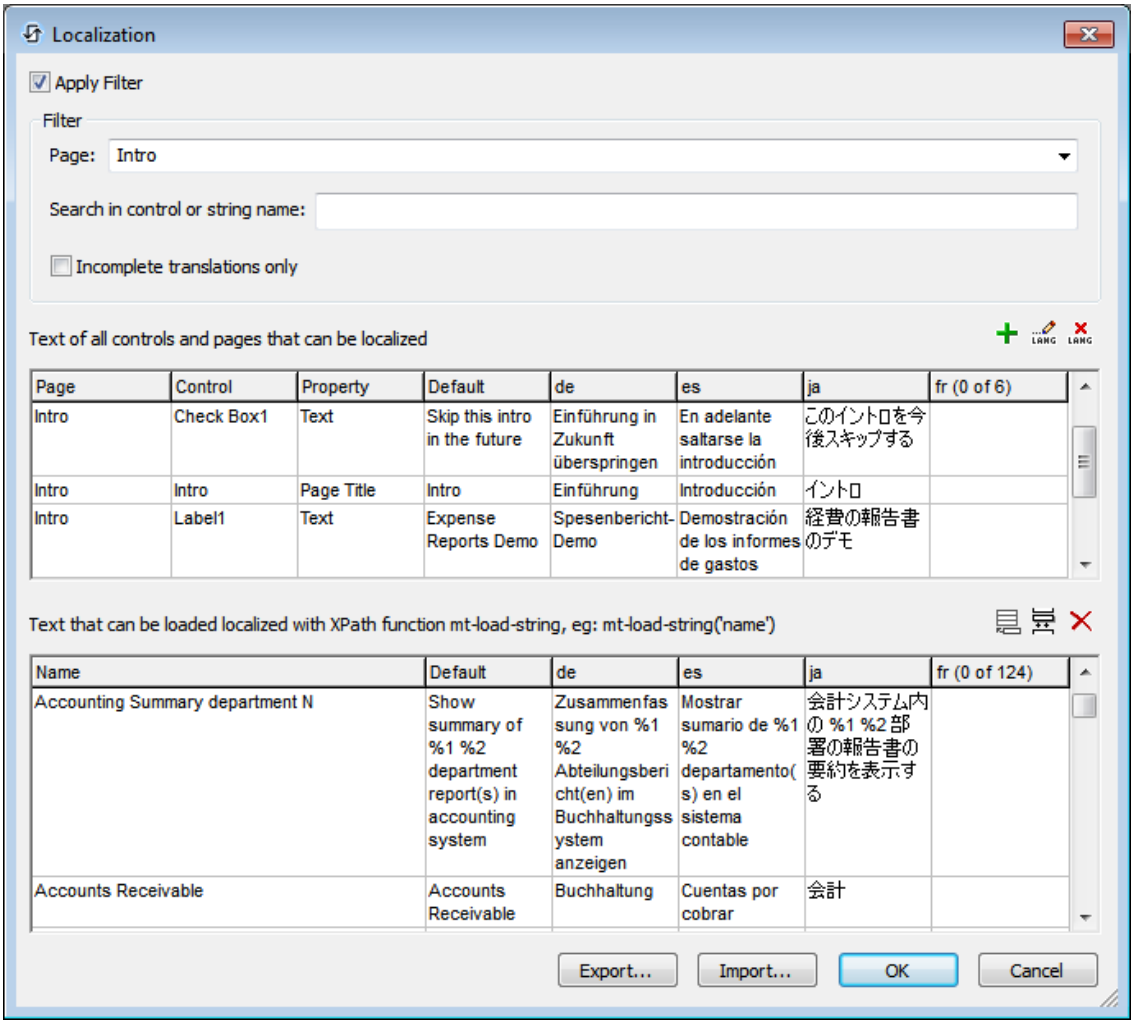
- *Re-deployment Timeout:* The time in hours after a new version of a solution is deployed that the superseded solution is kept on the server. This overlap time enables clients currently using the older solution to complete work. The default is 5 hours.
- *XPath Compatibility Mode:* When set to `true`, 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`.
- *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 refers to the embedding of data sources in the project (design) file. If *Automatic Re-embedding* is enabled (`true`), then page data 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`.
- *Top Level Margins:* Top-level controls are controls that are located directly inside the design—that is, all controls that are not inside a table. The margin you set in these

options for top-level controls 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 these page properties 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.)

6.5 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**. Additionally, named text strings can also be localized and subsequently referenced anywhere in the design by using the `mt-load-string` extension function in an XPath expression: **`mt-load-string('NameOfString')`**.



Localized solutions can be tested by selecting the simulation language you want via the **Project | Simulation Language** command and then running a simulation.

6.6 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 (*screenshot below*) contains all the namespaces that have been declared for the project, irrespective of what page is currently active in Page Design View.

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 (*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 (*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.

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.

6.7 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.

6.8 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.

- Embed XML in Design File
- Data Querying with XQuery 3.1
- Data Storage on Servers
- Persistent Data Storage on Clients

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**.

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 to create and check XQuery expressions.

Data Storage on Servers

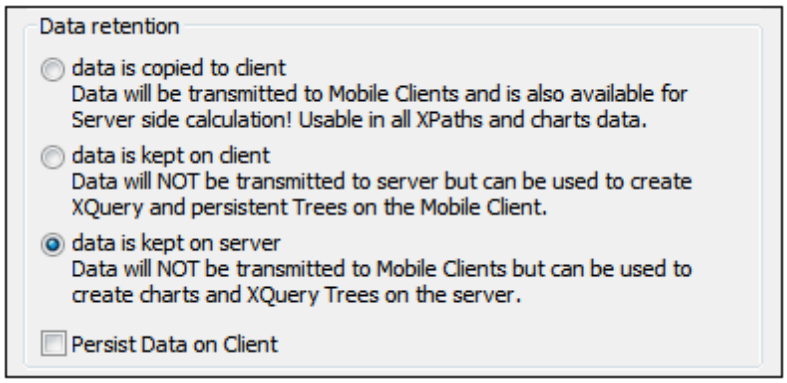
The speed with which data for solutions is processed can be enhanced by storing certain types of data on the server:

- 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 (Screen 2). It is also available as a command in the context menus of root nodes in the Page Sources Pane.



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, 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.

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*: 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*: In the Page Sources Pane, 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*: This setting can be defined in the Styles & Properties Pane. 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.

Chapter 7

Pages

7 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. The pages of a project are managed in the Pages Pane. For an overview of pages, see the subsection Pages of a Project.

The design components of a page—typically the controls that are added to the page—process data that comes from specified data sources. These data sources are defined in the Page Sources Pane. After the data sources have been defined, they can be used by the controls and actions of that page. For an overview of how these "page data sources" are to be used, see the subsection Data Sources of a Page.

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.

An important feature of pages is that actions can be set to execute when page-related events are triggered. For example, every time a page is loaded, its data sources can be updated. The various page events and their settings are described in the subsection Page Events.

In this section

This section is organized into the following subsections:

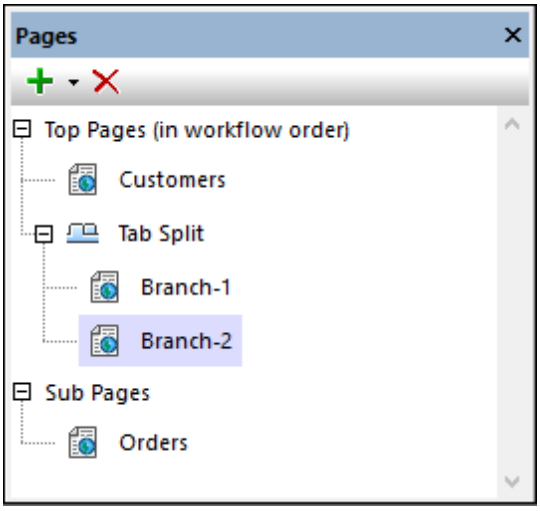
- Pages of a Project
- Data Sources of a Page
- Page Properties
- Page Events

7.1 Pages of a Project

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

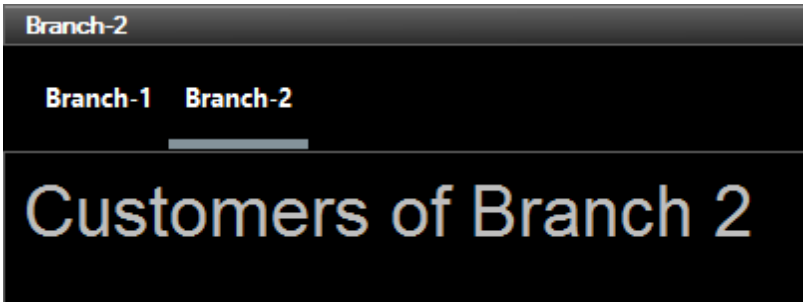
The pages of a project are managed in the Pages Pane (*see screenshot below*). You can add three types of pages to your project: (i) top pages, (ii) tabbed pages (tab splits), and (iii) sub pages. Do this via the **Add** icon of the menu bar, or via the context menu of individual pages (right-click a page to see its context menu). The top-down order in which the pages are listed in the Pages Pane 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.

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.

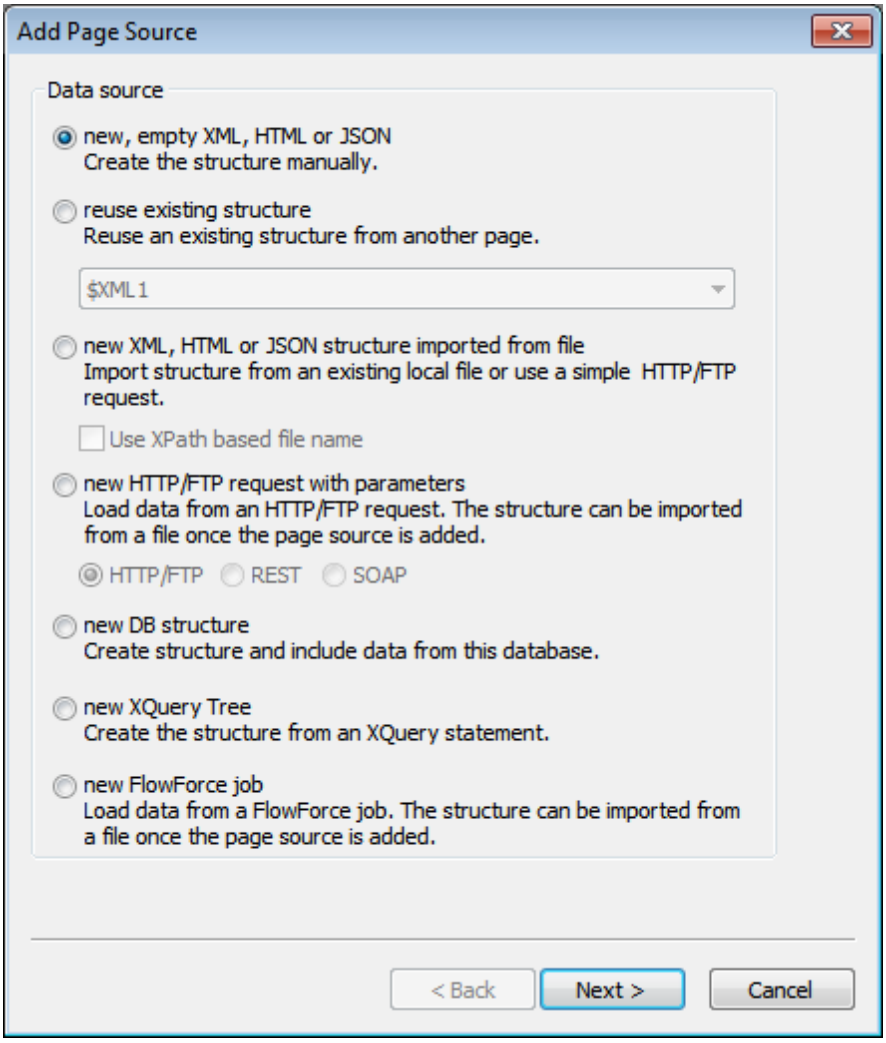


Sub pages

Sub pages are added via the **Add** icon in the menu bar of the Pages Pane, 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 action (for example, on a button control). 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 action to go to any page in the design. For more information about managing pages, see the description of the Pages Pane.

7.2 Data Sources of a Page

Every page of a design has a set of data sources, which are managed in the Page Sources Pane. To add a data 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 data source you want to add. If you want to add the same data source as one that you have used in an already existing page, then select the *Reuse existing structure* option and select from the existing data sources that are displayed in the combo box.



The second page of the Add Page Source dialog enables the following options to be set: (i) whether the data source is editable or not, when data is loaded from and saved to the data source, and whether data is stored on the client or server. For details of these data source options, see Page Source Options.

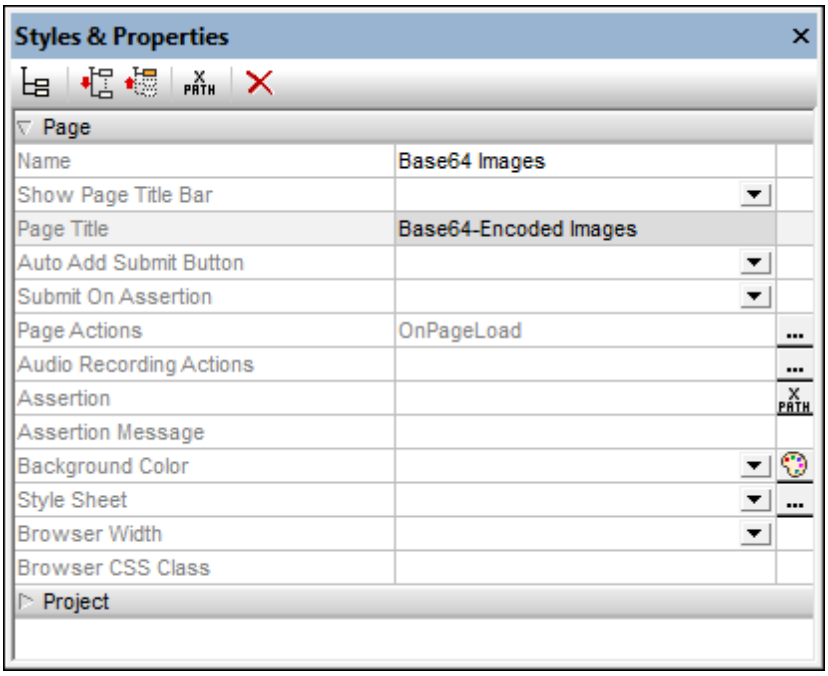
A number of settings for data sources and their individual nodes can be set separately. For a full listing of the context menu commands of different types nodes, see Context Menus (of Data Sources).

For more detailed information about data sources, see the sections Page Sources Pane and Data

Sources.

7.3 Page Properties

Page properties are defined in the Styles & Properties Pane and are described below. The screenshot below shows default values.



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.

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, 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 and Actions 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.

▼ 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 control. (If there are multiple Assertion Message 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 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 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 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

---

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the project. 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*). See the section Style Sheets for more information.

▼ Browser Width

Specifies the page width on browsers, either as a percentage of the screen width or as an absolute pixel value. Select a value from the dropdown list. The default value is calculated automatically by the browser.

▼ 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) to assign properties for this control separately.

7.4 Page Events

Each page in a MobileTogether design has certain predefined **events** associated with it:

- `OnPageLoad`: You can define a set of actions to execute when the page is loaded.
- `OnPageRefresh`: 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`: Actions to execute when the **Back** button of the solution is tapped/clicked.
- `OnSubmitButtonClicked`: Actions to execute when the **Submit** button of the solution is tapped/clicked.
- `OnServerConnectionError`: 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` variable to debug such errors. For an overview of how this event can be used, see the section Server Connection Errors.
- `OnEmbeddedMessage`: An action that generates the message that is posted from an IFrame embedded in a webpage to the workflow on the server. See Embedded Webpage Solutions for a description of this feature.

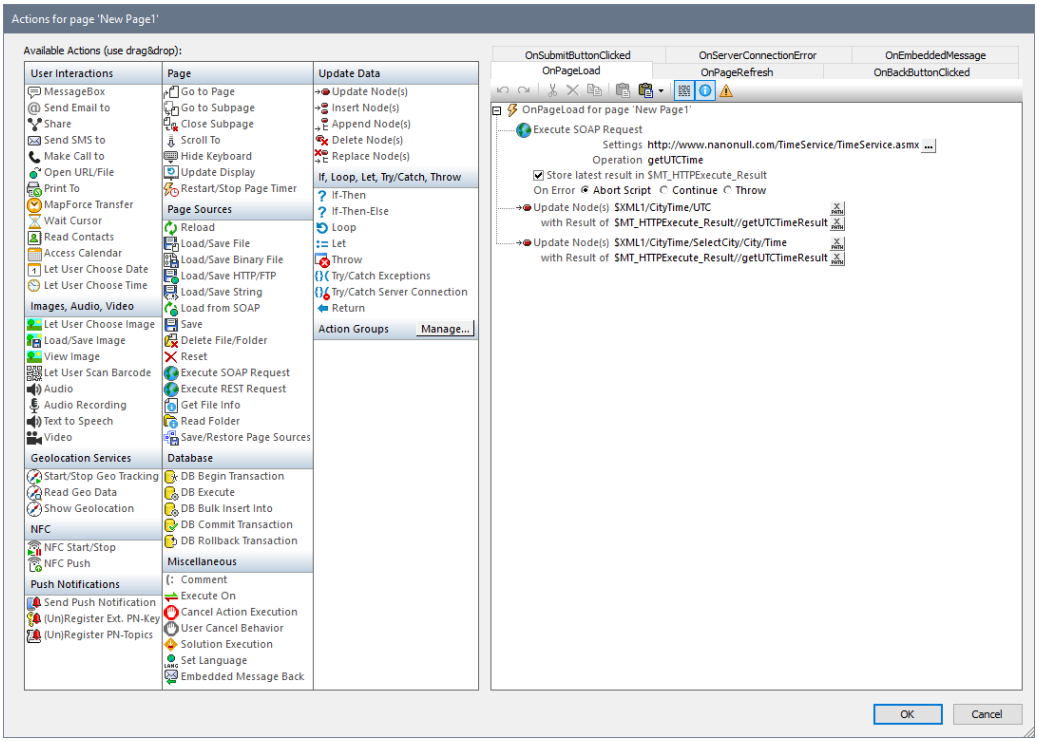
These events are called **page events** (as opposed to control events), and each can have one or more **actions (page actions)**. When a page event is triggered, the page action that is defined for it is carried out.

Defining the actions of a page event

To define the actions of a page event, access the Page Actions dialog (*screenshot below*). Select the tab of the event you want, then define its actions.

You can access the Page Actions dialog (*screenshot below*) in one of the following ways:

- Click **Page | Page Actions**
- Right-click anywhere in the page, and select **Page Actions**
- In the Styles & Properties Pane, go to the *Page* section. and click the **Additional Dialog** button of the `Page Actions` property
- Click **Page | Actions Overview** to display the Actions Overview dialog. Then click the **Additional Dialog** button of the page event you want to define



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 and on the different types of available page actions, see the section Actions.

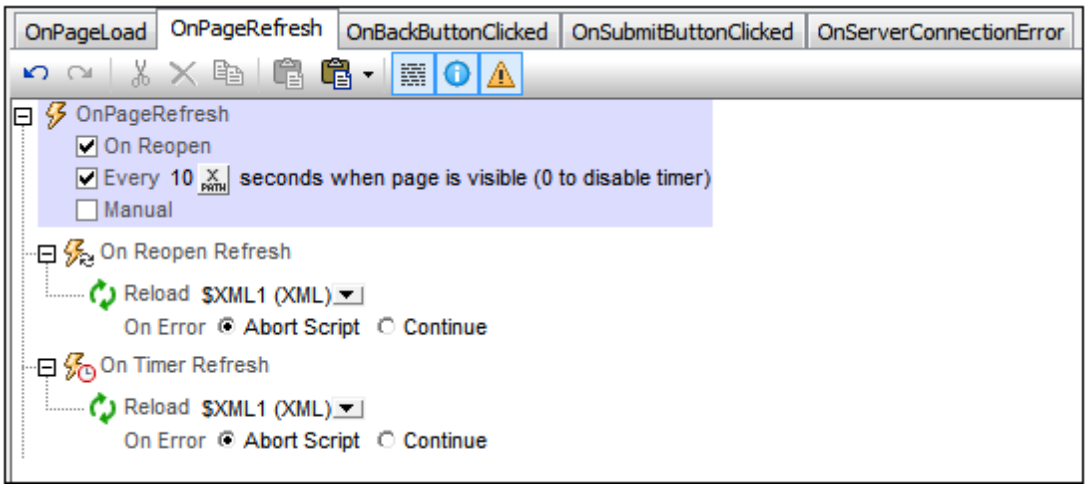
OnPageLoad

The `OnPageLoad` event is available for all pages (top pages and sub pages). 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.

OnPageRefresh

The **`OnPageRefresh`** event is available for all pages. 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.



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* and the `SolutionExecution` 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 data 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 action to the event that changes the refresh interval. *See the SOAP Requests tutorial 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 action to a suitable event.

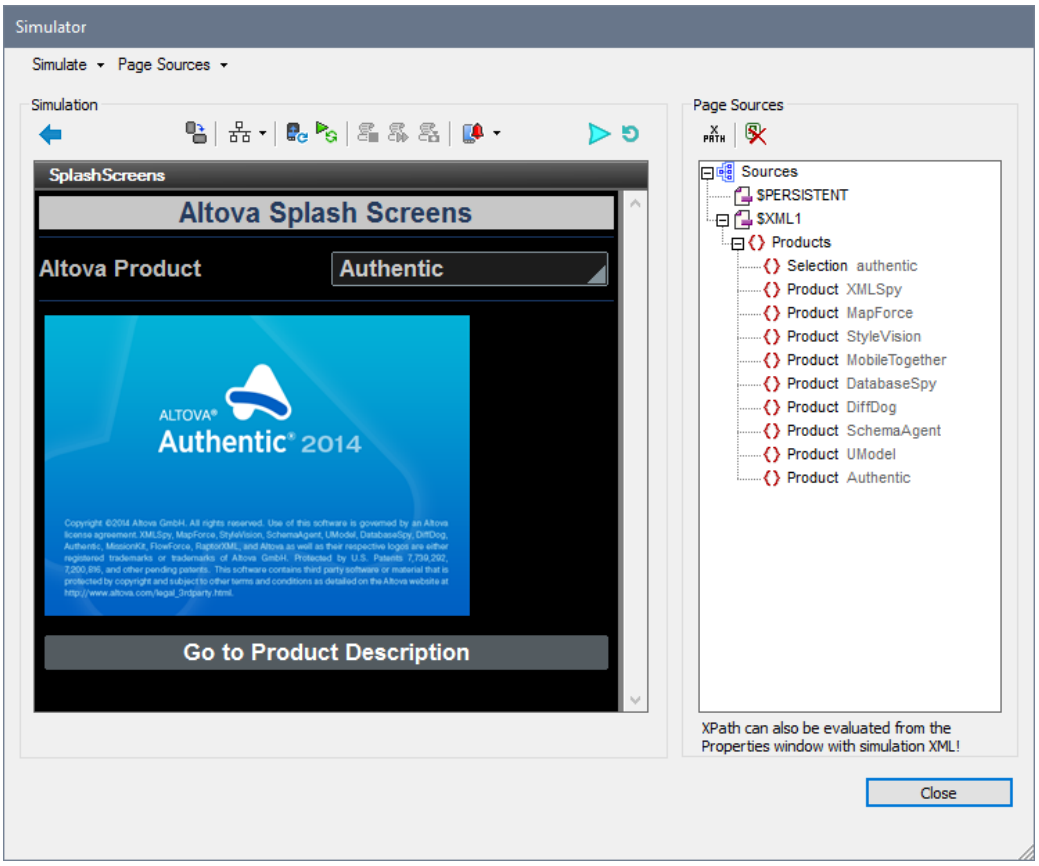
On Manual Refresh
If this option is selected, the page will display a **Refresh** button. When the user taps/clicks this button (or, on iOS, pulls down to refresh), the *On Manual Refresh* actions are executed. *See the SOAP Requests tutorial for an example*.

For an example of page refreshes, see the *SOAP Requests tutorial*.

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.



OnBackButtonClicked

The `OnBackButtonClicked` event is available for all pages (top pages and sub pages). 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:

- *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.

OnSubmitButtonClicked

The **Submit** button appears at the top right of all Top Pages—if the Auto-Add Submit Button property of the top page has been left at its default value of **`true`** in the Page Properties 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.

OnServerConnectionError

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:

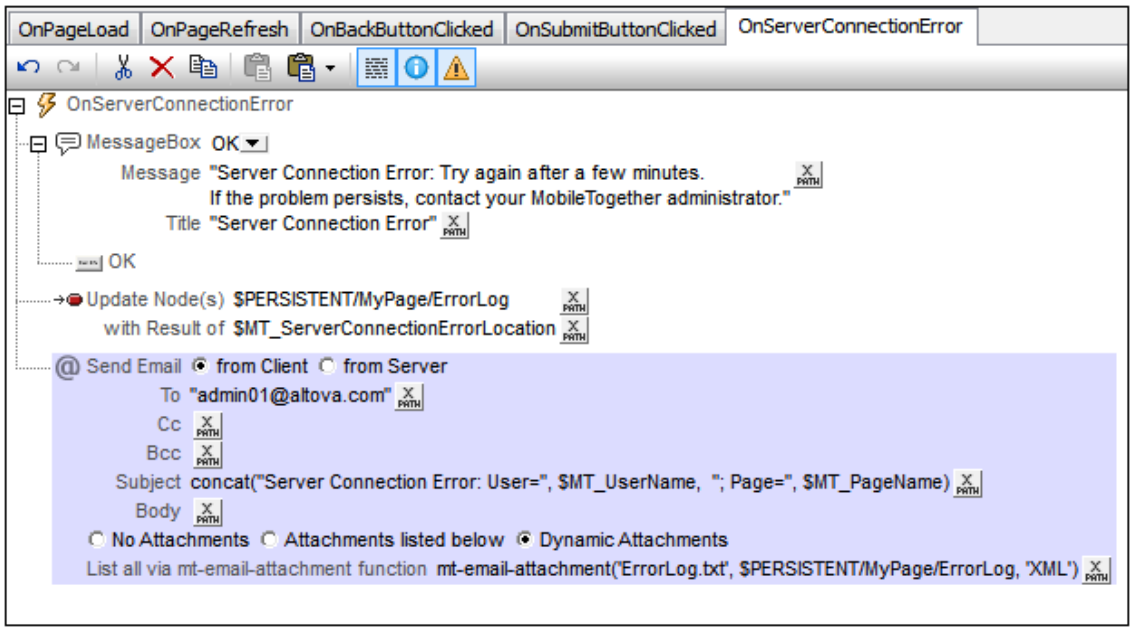
- a suitable error message to the end user (client device), and
- the subsequent actions to take.

You can also simulate a connection error in the Simulator.

Note: When a server connection error occurs, the first of the following actions that exists is triggered: (i) a Try/Catch Server Connection Errors action, (ii) action/s for the **`OnServerConnectionError`** event (this event), (iii) a MobileTogether message about the error, following which the workflow is resumed.

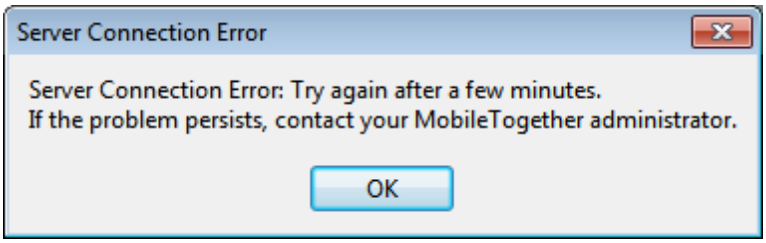
Defining what to do when there is a connection error

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 `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.



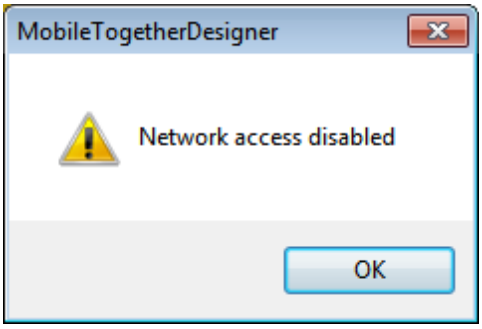
The screenshot above defines a sequence of three actions to perform:

1. Send an error message to the client (*screenshot below*).



2. Use the `MT_ServerConnectionErrorLocation` variable to save the action stack that triggered the `OnServerConnectionError` page event. (The variable should be used for debugging purposes; see `MT_ServerConnectionErrorLocation` for details.) Alternatively to the `MT_ServerConnectionErrorLocation` variable, you can use the `Update Node` 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.

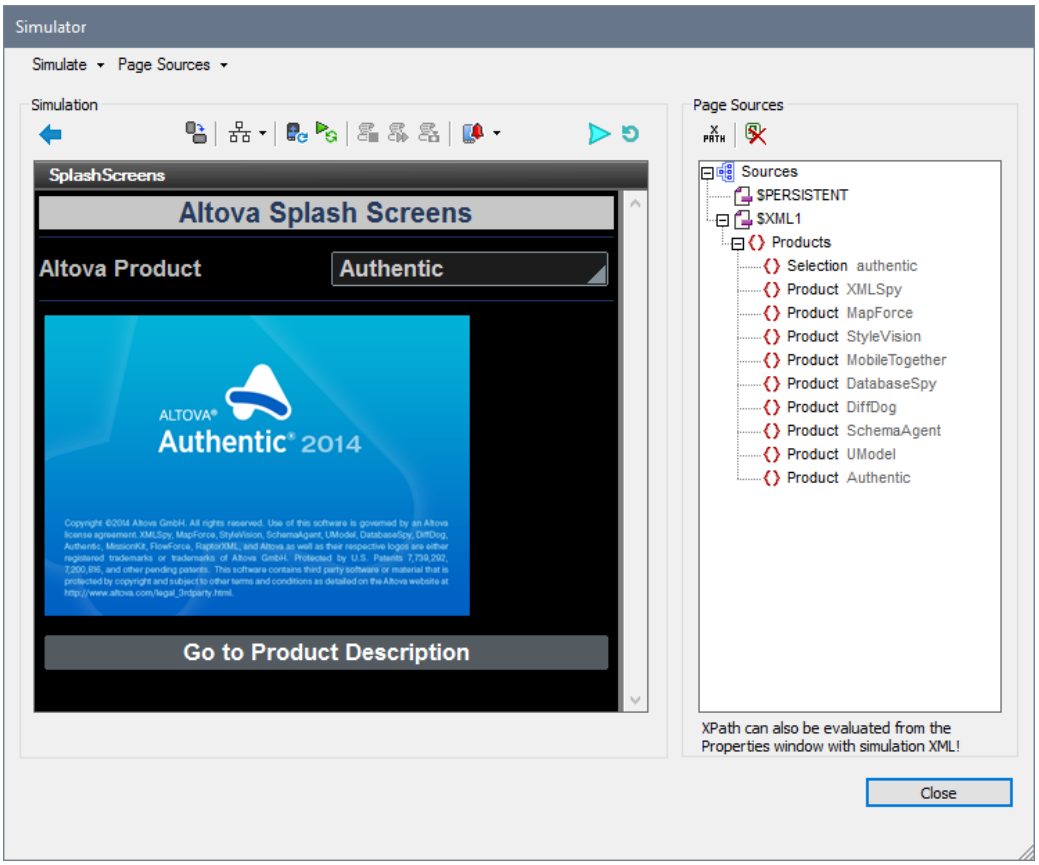
Note: If no action is defined in the tab of the `OnServerConnectionError` page event, then a generic *Network access disabled* message is sent to the mobile device (*screenshot below*):



Simulating a server connection error (for testing)

You can use simulations in MobileTogether Designer and simulations on the Server to test the behavior of a solution. Do this as follows:

1. Start the simulation (for example, with **F5**). The simulator starts (*screenshot below*).



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` page event are triggered.
4. To enable server access again, click **Enable Server Access** in the simulator.

OnEmbeddedMessage

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 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 **`$MT_EMBEDDEDMESSAGE`** JSON page source is automatically created if at least one action is defined for the event handling of `OnEmbeddedMessage`. If you wish to carry out no other action except to create the page source, then add an action that does not interfere with the workflow, for example, a Comment action. If you do not add an action, then the **`$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. 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

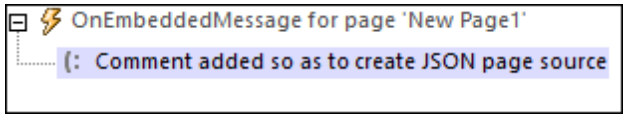
for the event, then these are executed.

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.

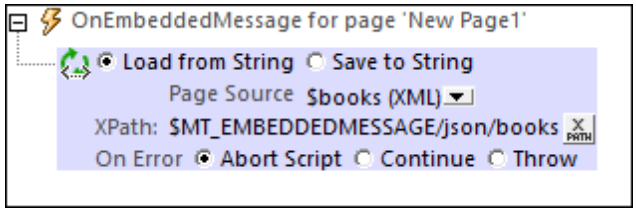
Two scenarios arise depending on the format of the source data in the webpage:

- *JSON format:* The JSON page source **`$MT_EMBEDDEDMESSAGE`** can be used directly in the solution, and data from this page source can be passed back to the webpage. In this case, any non-intrusive action (such as the Comment action) can be defined for `OnEmbeddedMessage` (*see screenshot below*). This activates event handling and suffices to create the **`$MT_EMBEDDEDMESSAGE`** page source.

  See Sending/Receiving JSON Data 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 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.

  See Sending/Receiving XML Data for an example of how to work with XML data.

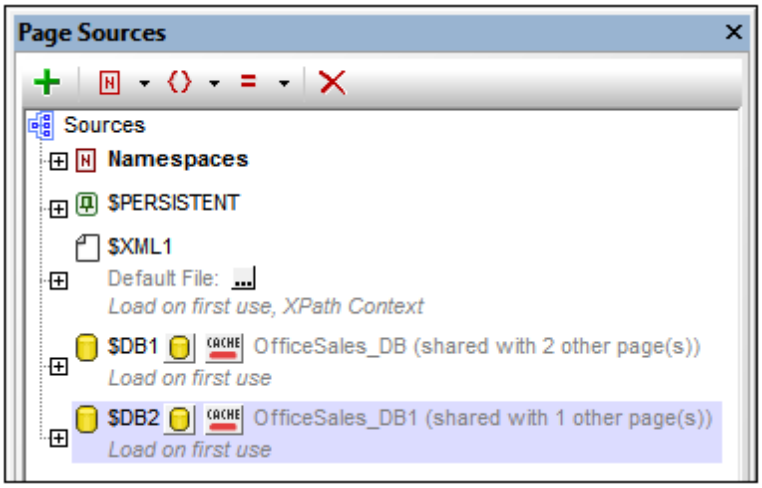
Chapter 8

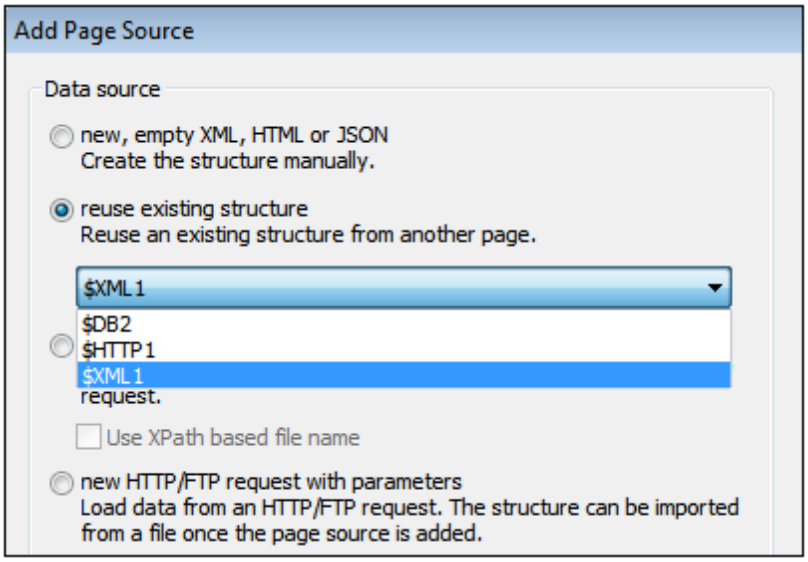
Data Sources

8 Data Sources

Each page can have a set of page data sources (or page sources). These sources (*see screenshot below left*) are available to controls of that page, and that page only. However, a single page source can be reused among multiple pages.

- New page sources are added via the Add Source command of the Page Sources Pane. How to add a new page source is described in the section Adding Page Data Sources.
- A page can reuse the page sources of other pages. An option for specifying the page source to reuse is available when adding a new page data source (*screenshot below right*).





Note the following points:

- **Page sources** are added in the Page Sources Pane (*screenshot above*), one source at a time.
- Page sources are added from **data sources**, which may be XML, HTML, JSON, XQuery, or DB data sources, data sources accessed via HTTP, FTP, REST or SOAP, or other page sources of the project. See the section Types of Data Sources for more information.

- Each page can have multiple page sources based on data sources of different kinds. For example, the screenshot above left shows a page that has one XML-based and two DB-based page sources.
- A page source can be editable or non-editable (read-only). This property is specified at the time when the page source is added and can be modified with the root node's context menu toggle command **Read Only Data**.

Organization of this section

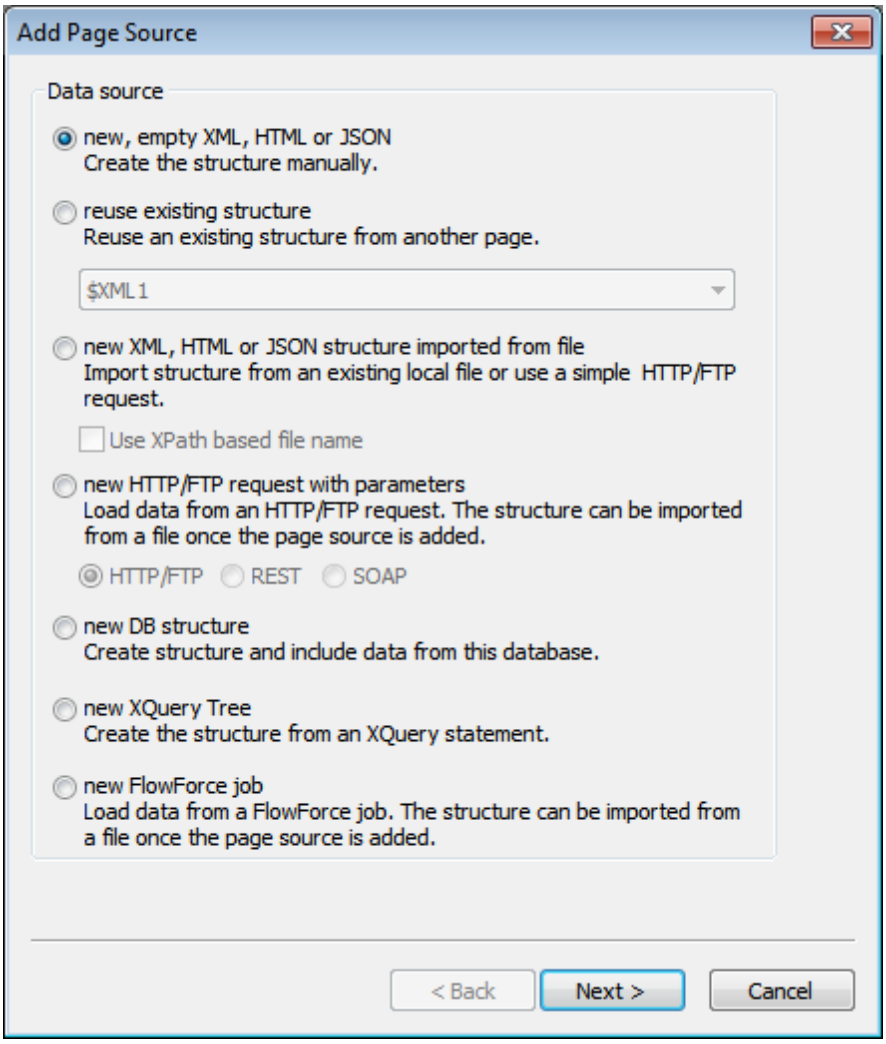
This section is organized as follows:

- Adding Page Data Sources: shows how to add a page source
- Types of Data Sources: lists and describes the various type of data sources that can be added to a page
- Page Source Options: describes the properties of page sources
- HTTP/FTP, REST, and SOAP Requests: explains the settings for making these requests
- Root Nodes: explains the concept of root nodes and root elements, and how these are used in the trees of data sources. This section also gives the naming convention of root nodes
- Page Source Trees: shows how the structure of a source tree and data in the tree's nodes are used in a page
- Namespaces in the Project: explains the significance of namespaces in the project, and how namespaces are used
- Caches: describes how the mechanism for caching data on the server works
- Context Menu: describes the context menu commands of the Page Sources Pane

8.1 Adding Page Data Sources

To add a page data source (or page source), select the page in the Pages Pane, and then do the following.

Click the **Add Source** icon in the toolbar of the Page Sources Pane to display the Add Page Source dialog (*screenshot below*). Select the type of data source to add (*listed below*), specify the properties of this data source, and click **Next**. The various types of data sources are described in the next section, Types of Data Sources.



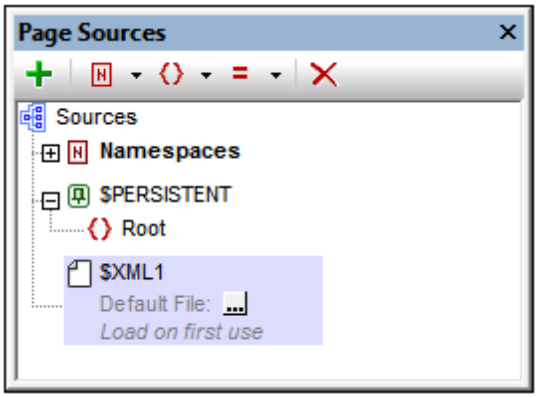
8.2 Types of Data Sources

The types of data sources that are available for addition as page sources are listed below:

- New empty XML, HTML or JSON

  When a new empty XML, HTML, or JSON page source is added in the Add Page Source dialog, it is created as an XML page source. You must specify, in the next screen of the Add Page Source dialog, whether the data source is an XML, HTML, or JSON file. A root node named **$XML** is created for the new page source (*see screenshot below*). If the data source is specified as XML or HTML, then no other node or any namespace is created in the tree. If the data source is specified as JSON, then a root element called **json** is created under the root node. You can now construct an XML document structure under the root node by using one or both of the following methods:

  - adding element and attribute nodes to the root node (with the toolbar commands to do this). See Tree Structure for details.
  - importing a structure from an XML, HTML, or JSON file via the **Import** command of the root node's context menu.

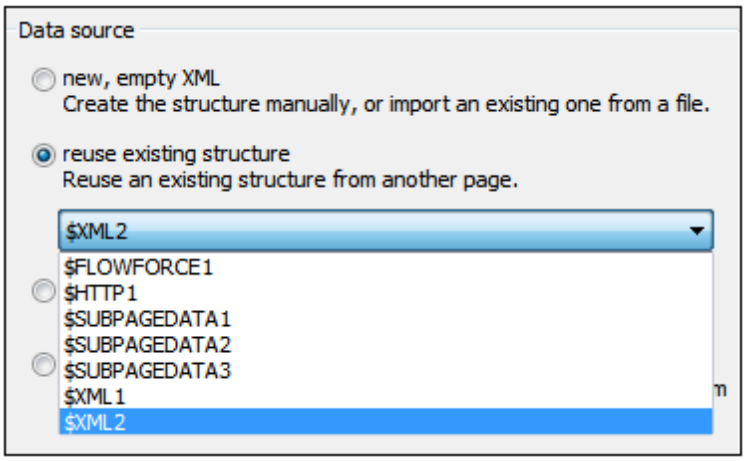


  How to add data to the tree (including by assigning a default file) is described in the section, Tree Data.

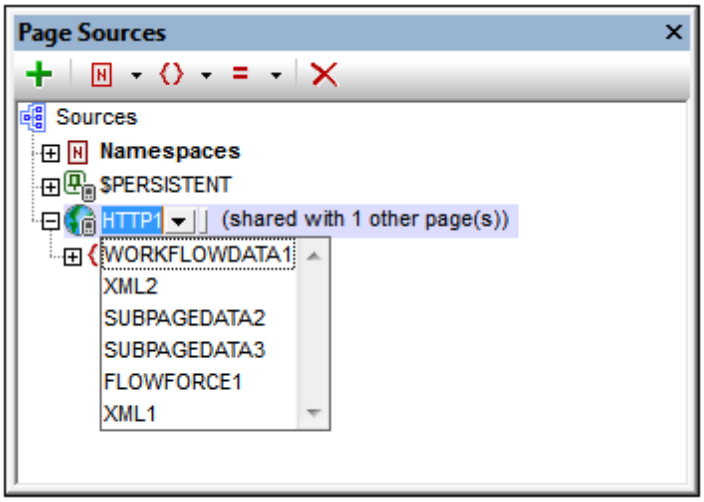
  Note: You can change the data type of the page source via the root node's context menu command **Data Type**.

- Reuse existing structure

  This option is useful if you want to reuse a structure that you have already created in another page of the project. The reusable page source must be in *another page*—not the same page—of the project, and the page could be either a top page or a sub page. The *Reuse* option is enabled only if a page source exists in another page of the project.



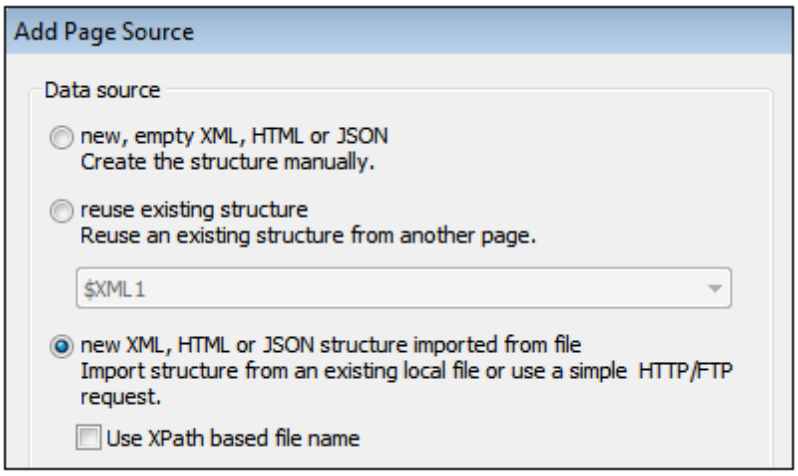
The available page sources are listed by the names of their root nodes in the dropdown list of the options's combo box (*see screenshot above*). Select the page source you want to reuse and click **OK**. A new root node is created with the same name and structure as the reused page source (*see screenshot 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. How to modify the tree structure is described in the section Tree Structure.

- New XML, HTML or JSON/JSON-5 structure imported from file

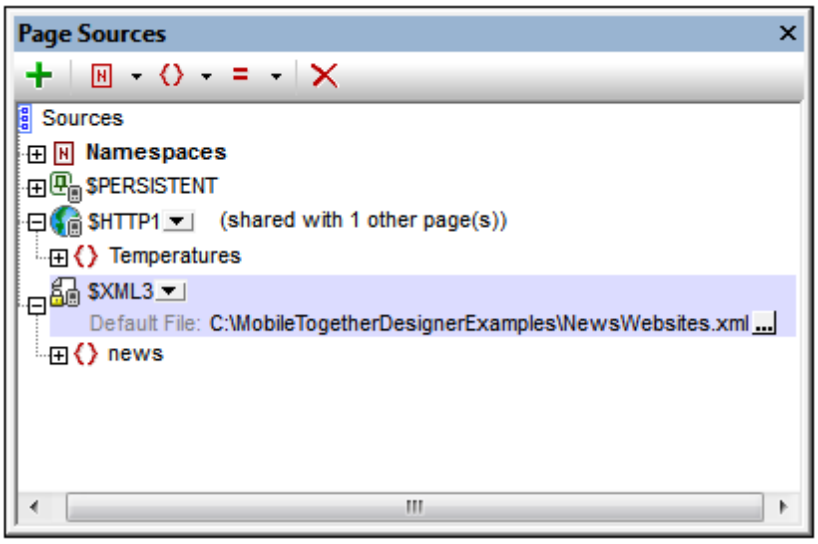
  If this option is selected (*see the screenshot below*), clicking **Next** displays the Add Page Source dialog, in which you set the usage options of the selected data source.



You must specify, in the next screen of the Add Page Source dialog, whether the data source is an XML, HTML, or JSON/JSON-5 file. (Hereafter, JSON is used as short form to refer to the set of JSON and JSON-5 documents.) What happens when you click the **Finish** button of this dialog depends on whether the *Use XPath-based file name* option (*see screenshot above*) was selected or not.

- If the check box was selected, the Edit XPath/XQuery Expression dialog appears. You can build an XPath expression here to generate the file URL you need.
- If the check box was not selected, a dialog box appears in which you can select the XML, HTML, or JSON file that provides the structure of the page source. You can browse for the file, use a file URL, or use a global resource.

The structure of the XML/HTML/JSON file is imported as the structure of the page source (*see screenshot below*). XML and JSON file structures are displayed under the **$XML** root node; HTML file structures are displayed under the **$HTML** root node. The structure of an HTML or JSON data source is imported as an XML tree structure. An imported JSON structure will have a root element named **json**. The XML/HTML/JSON (data 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.



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.

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 data source tree in the Page Sources Pane. For example:

```
<table>
   <tr/>
   <tr/>
</table>
```

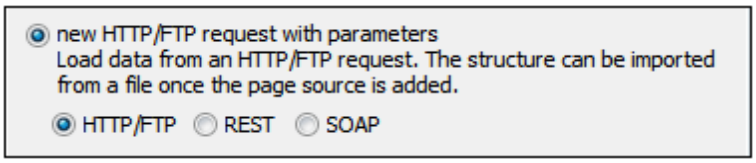
will be corrected to:

```
<table>
   <tbody>
      <tr/>
      <tr/>
   </tbody>
</table>
```

- New HTTP/FTP request with parameters

  A data 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 data 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, whether the data source is an XML, HTML, or JSON file. You can subsequently change your selection in the respective Settings dialog (HTTP/FTP or REST). (If you select SOAP, the data source must be parsed as XML; this option is automatically set and cannot be changed.)



On clicking the **Finish** button of the Add Page Source dialog, the Edit Web Access Settings dialog (for HTTP/FTP requests), RESTful API Request dialog (for REST requests), or WSDL File Selection dialog (for SOAP requests) is displayed. See the section HTTP/FTP, REST, and SOAP Requests 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) and data from the data 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.

▼ New DB structure

This option enables you to create a structure and add data from a database. On selecting this option and clicking **OK**, the database (DB) Connection Wizard appears. After making the connection to the DB, you can select the table data to be imported for data source structure and data content. A root node is inserted, with the database name next to the name of the root node. The root node also has the DB structure below it. See the Databases section for more information. The Database-And-Charts tutorial provides an example of how to use DBs.

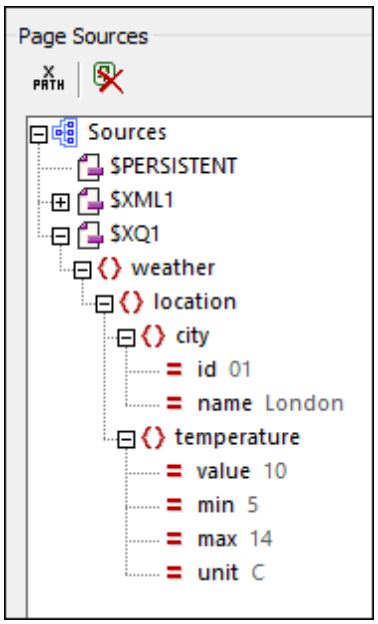
Note: If SQL statements are stored in a page source, they might trigger firewall rules while the design is executed on a client device. To prevent this, it is recommended that you do one of the following: (i) set the page source property *Keep data on* to `Server only`; (ii) use SSL for client connections; (iii) assemble the SQL statement on the server when required..

▼ New XQuery tree

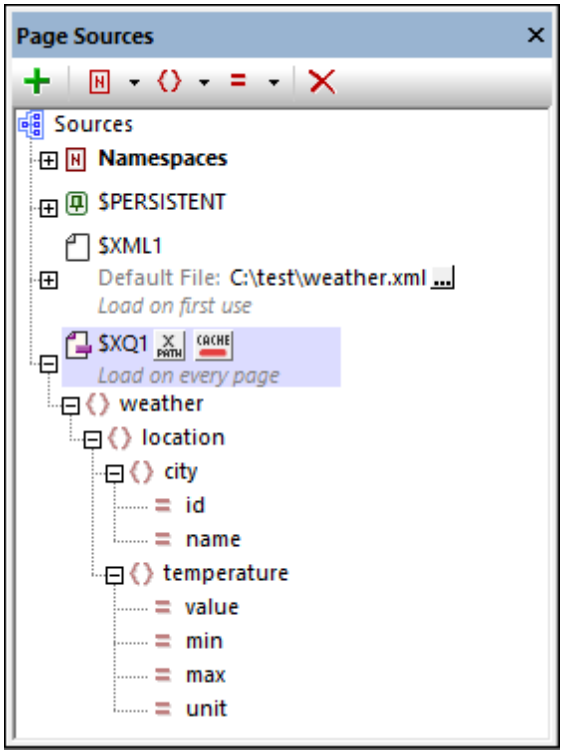
Selecting this option and clicking **OK** opens the Edit XPath/XQuery Expression dialog. Enter an XQuery statement that generates the required data structure (and optionally data), and click **OK**. A page data source with a root node named **`$XQ`** is created that has the structure specified in the 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"}}
}
}
```



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 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 data source's **Load Data** option that you selected (*On First Use*, *On Every Page,* or *Not Automatically*).

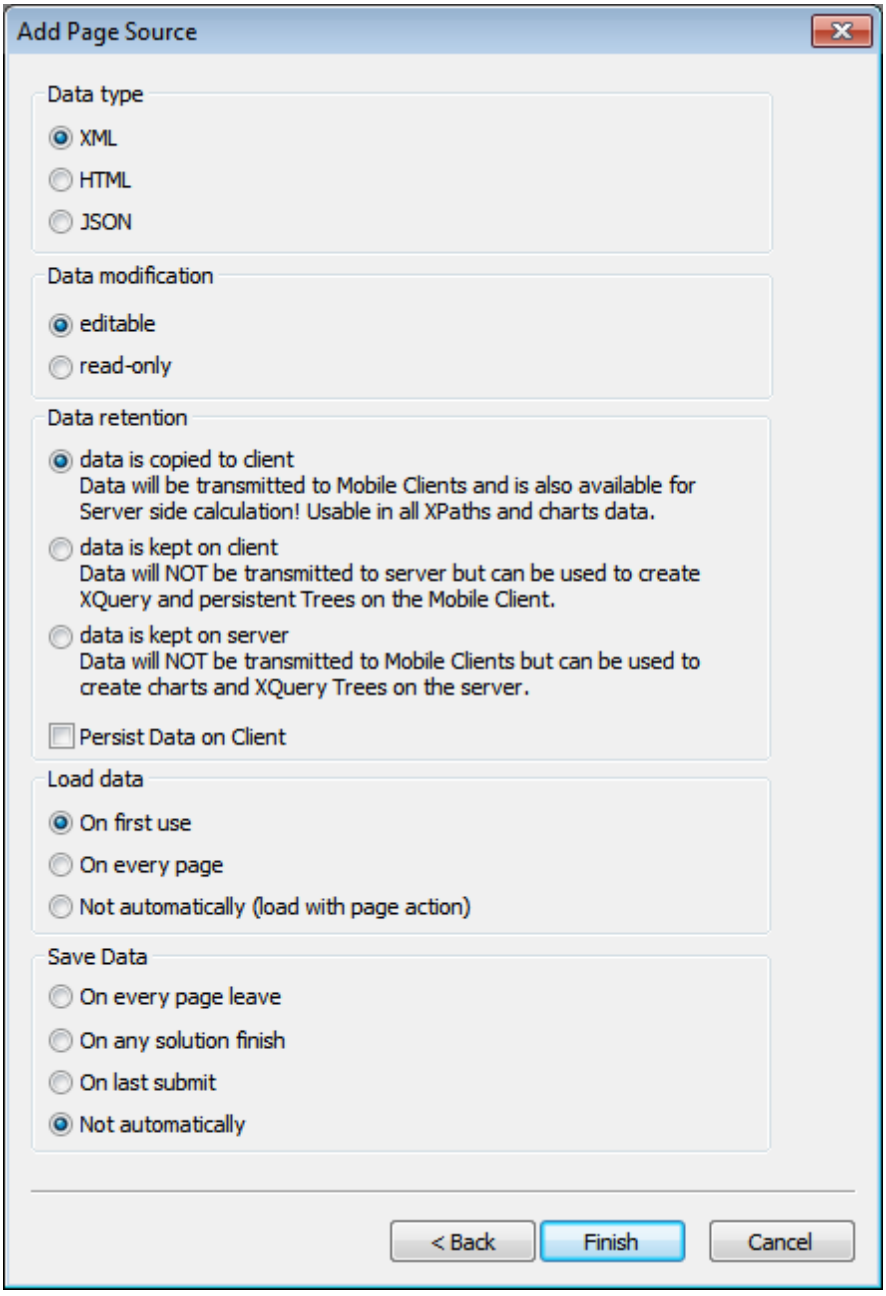


- New FlowForce job

  Selecting this option and clicking **OK** opens the Edit FlowForce Settings dialog. Enter the FlowForce settings to connect to a FlowForce server and specify a FlowForce job. On clicking **OK**, a new data source is created that contains data retrieved by running the FlowForce job. The structure of the data source can be modified subsequently. How to do this is described in the section, Tree Structure. For information about Altova's FlowForce application, see the FlowForce page on the Altova website.

8.3 Page Source Options

After you have selected the type of data source that you wish to add, and clicked **Next**, the Add Page Source dialog displays the usage options of the selected data source (*see screenshot below*). Select the options you want, and then click **Finish**.



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 data 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**). 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** project property, and the Save action.

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.

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 data 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*.

8.4 HTTP/FTP, REST, and SOAP Requests

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: In this situation, requests are typically made to load data from external sources
- When defining actions related to page sources: 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:

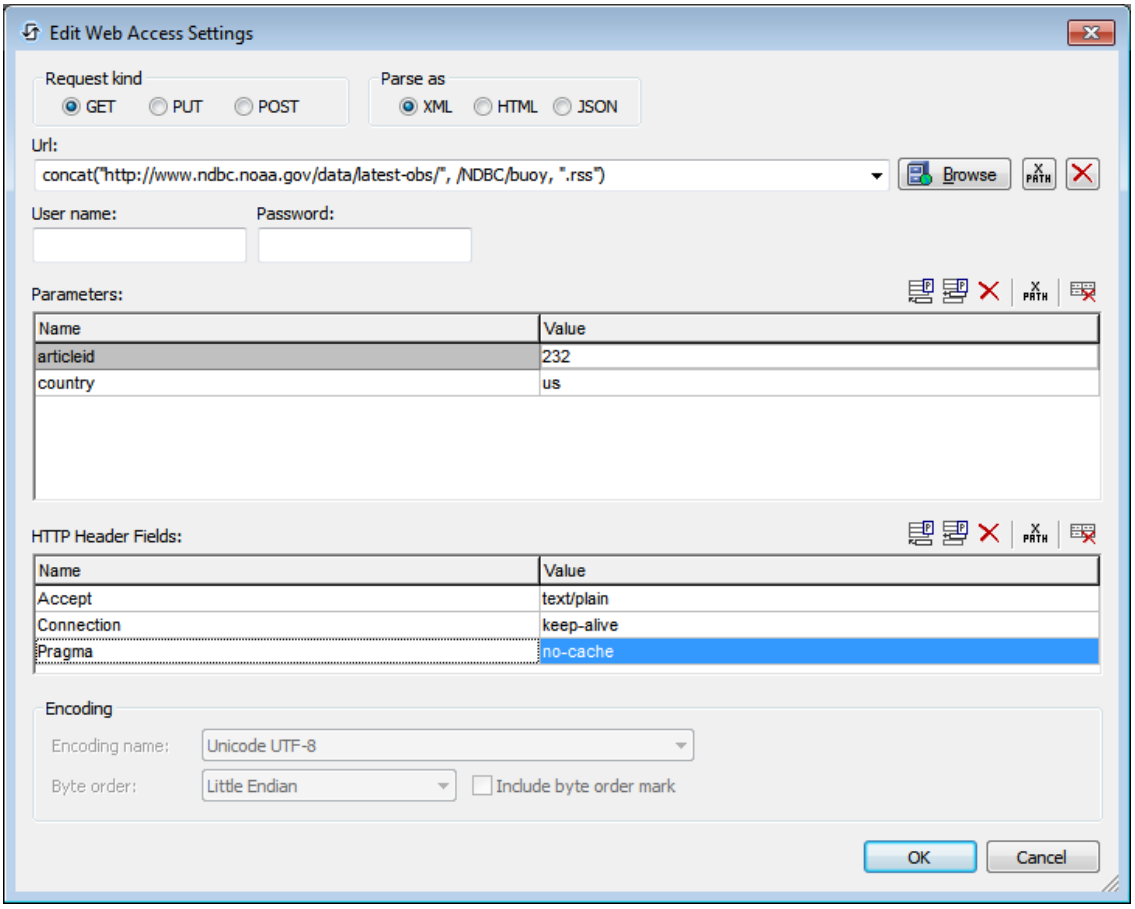
- HTTP/FTP request settings
- REST request settings
- SOAP request settings

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 for more information.

HTTP/FTP Request Settings

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.



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 data source tree in the Page Sources Pane. For example:

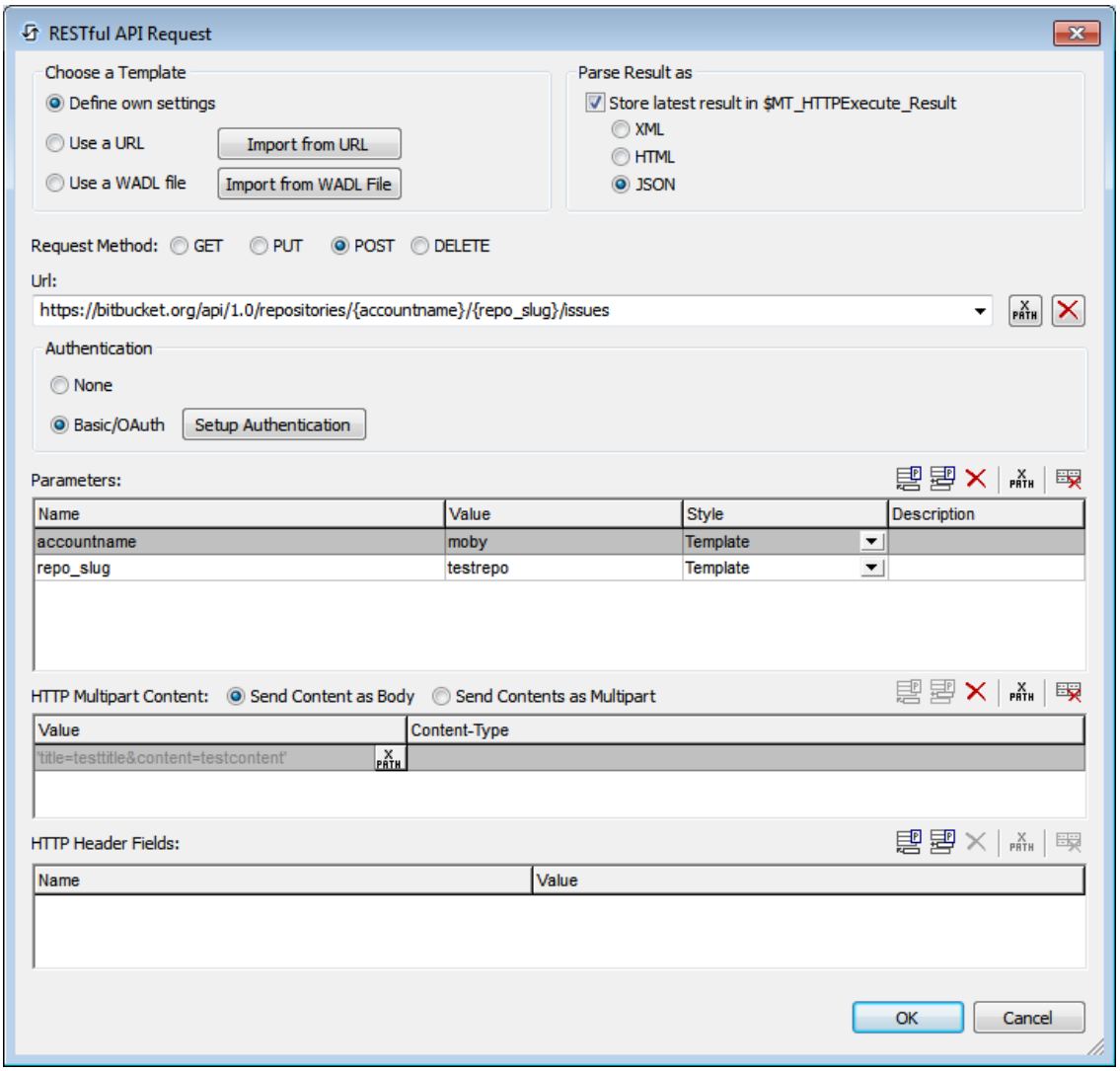
```
<table>
   <tr/>
   <tr/>
</table>
```

will be corrected to:

```
<table>
   <tbody>
      <tr/>
      <tr/>
   </tbody>
</table>
```

REST Request Settings

The settings for REST requests are defined in the RESTful API Request dialog (*screenshot below*). 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.



The various settings are described below:

- Templates and result parsing, including `$MT_HTTPExecute_Result`
- Request method
- URL
- Authentication
- Parameters
- HTTP content
- HTTP header fields

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):

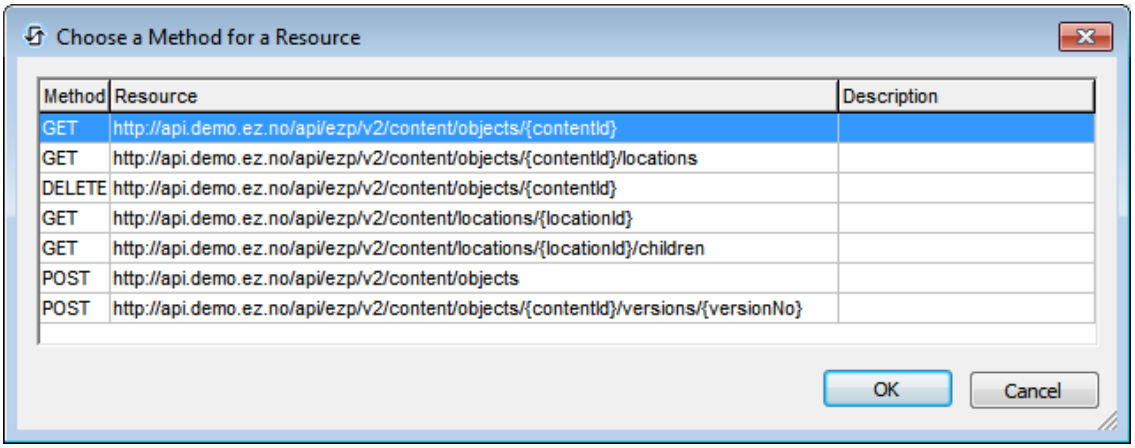
```
https://www.google.at/search?q=REST+WADL&ie=utf-8&oe=utf-
8&gws_rd=cr&ei=89cDVrDHMIP0Up_5vcAB
```

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 below.

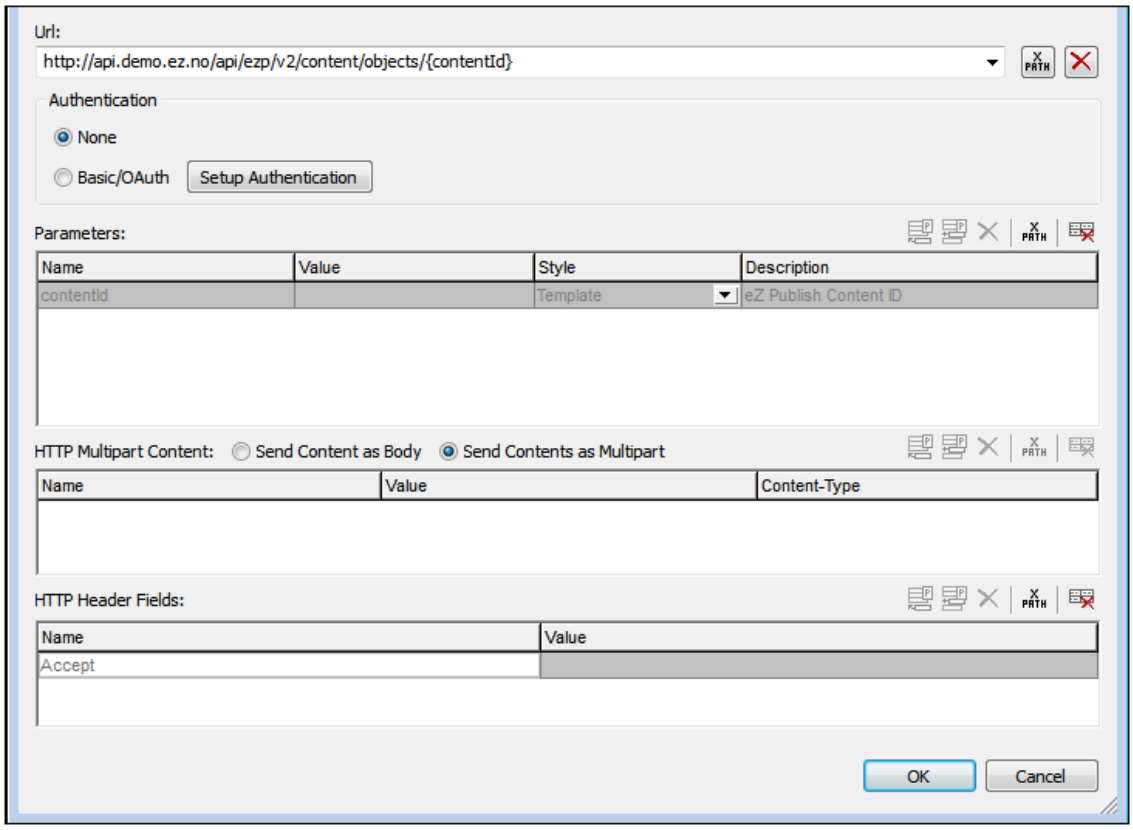
Use a WADL file

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.



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.



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.

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.

Request method

In the *Own Settings* and *URL* templates, you must specify the Request method (`GET`, `PUT`, `POST`, or `DELETE`). 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.

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.

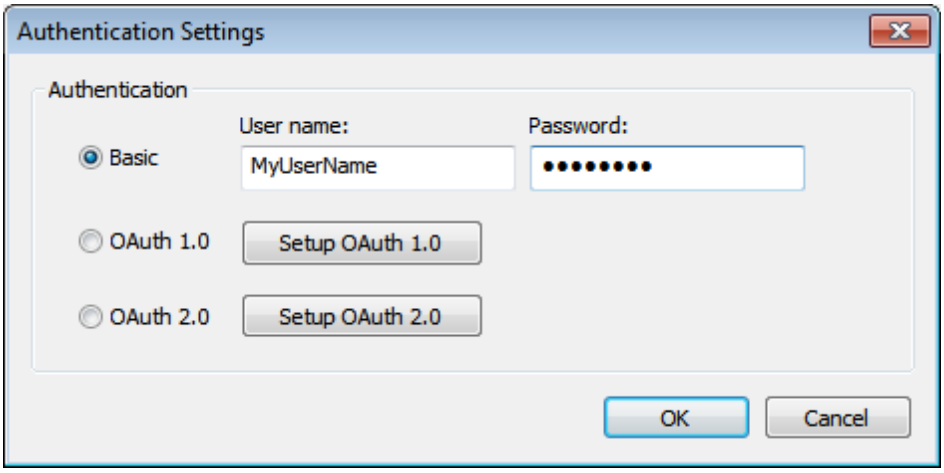
Authentication

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.



OAuth authentication

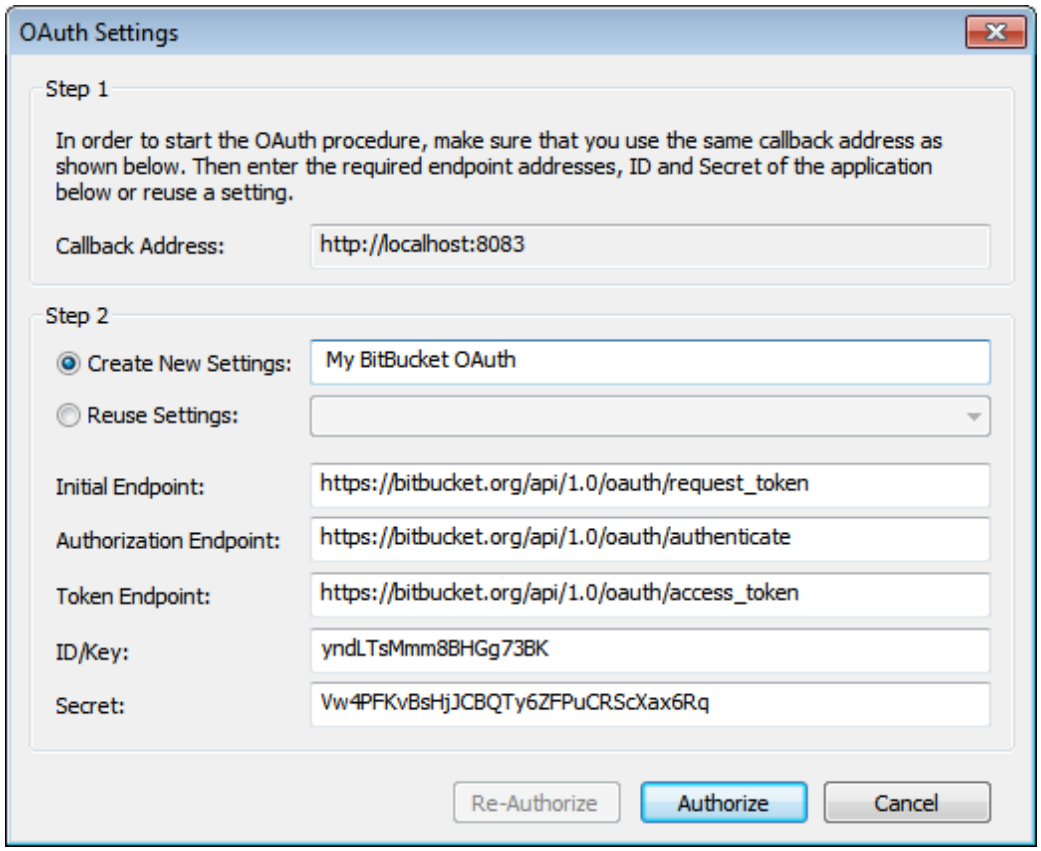
OAuth essentially authenticates MobileTogether Designer and authorizes access to the resources of the web service identified by the URL. This assumes that the web service supports OAuth. As an example web service that supports OAUTH, see the BitBucket documentation about OAuth 1.0.

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) and 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.



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

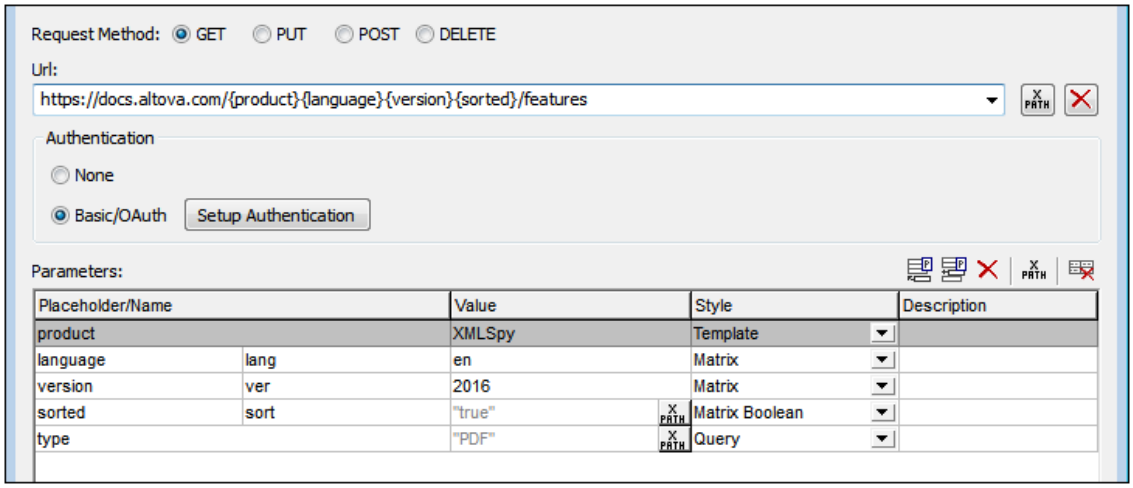
In the *Define Own Settings* template, parameters can be freely added, edited and deleted. In the *Use URL* and *Use WADL* templates, however, you can only edit the values of parameters; you cannot add or delete parameters, or edit their names.

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`.



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

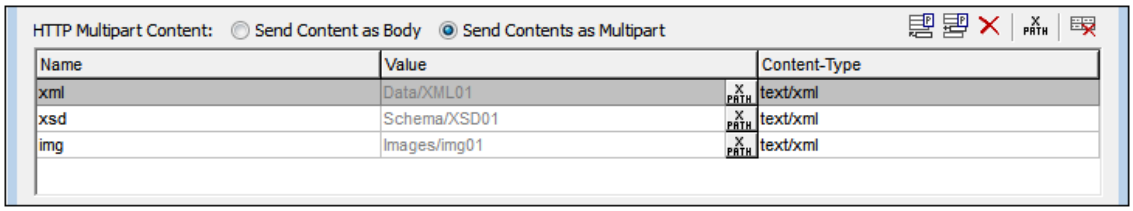
At the top right of the Parameters table, there are icons that enable you to manage entries in the

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

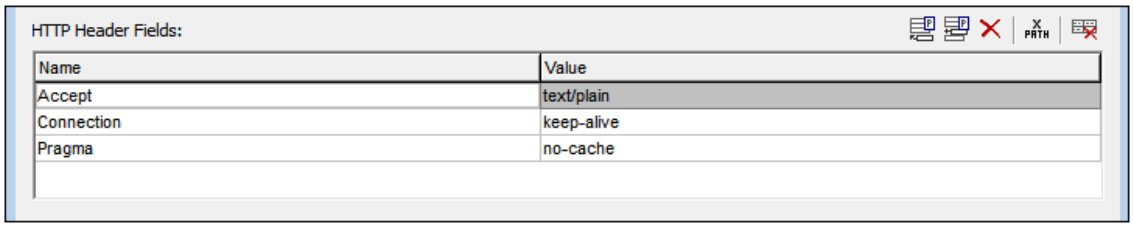
You can specify content to send with HTTP PUT and POST requests. You can send the content as a single item in the body of the request (*Send content as body*) or as multiple items in a multipart request (*Send contents as multipart*). Select the appropriate radio button from these two options.



Append or insert a content entry using the icons at top right. Then enter a name for the content part being sent, and the content part's content type. 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.

HTTP header fields

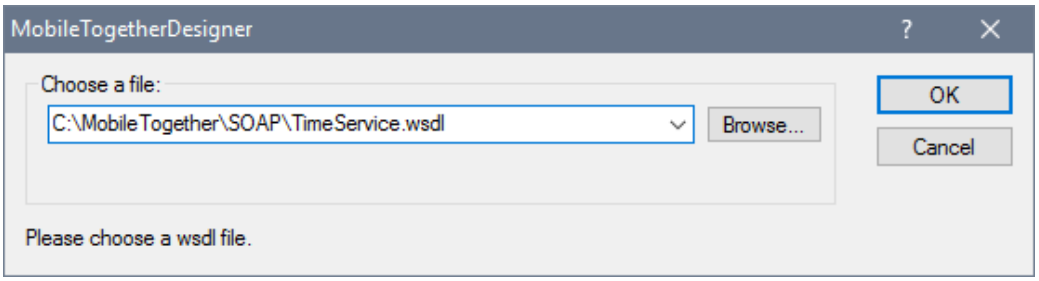
HTTP header fields are colon-separated name-value pairs, for example: Accept:text/plain. Append or insert an entry for each header, and then enter the header name and value (*see screenshot below*).


HTTP Header Fields:
Name
Value
Accept
text/plain
Connection
keep-alive
Pragma
no-cache

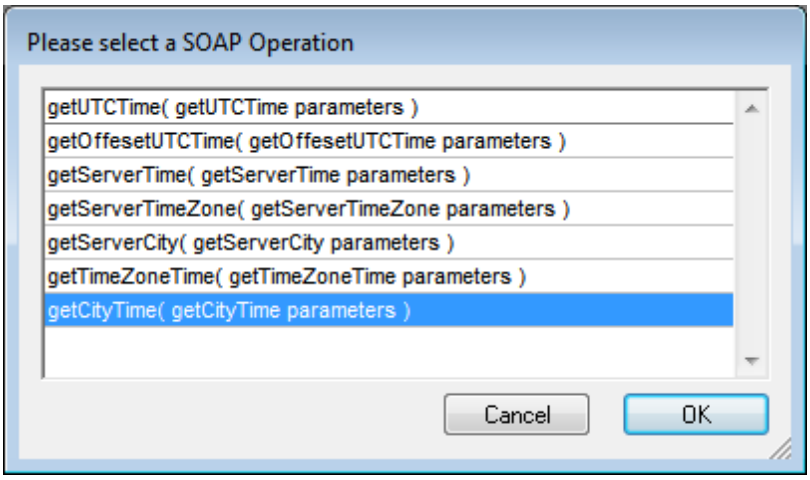
SOAP Request Settings

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:

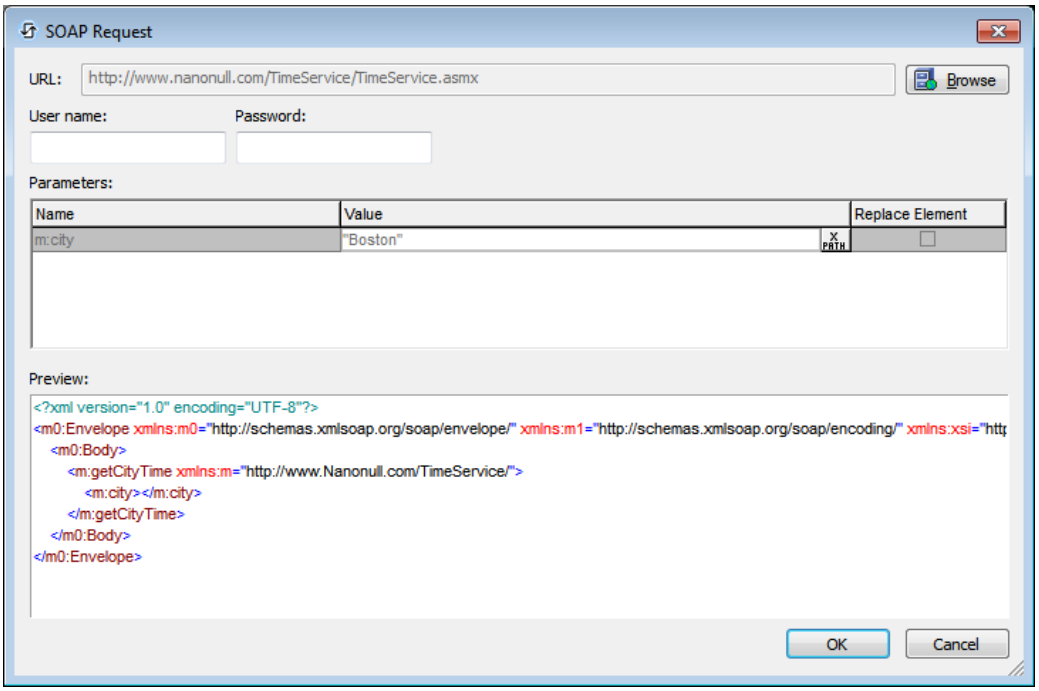
1. Add a page source by clicking the **Add Page Source** icon in the Page Sources Pane. Select *New HTTP/FTP request with parameters*, and then select its *SOAP* radio button. Complete the page source settings in the next screen and click **Finish**.
2. A dialog appears that prompts you to browse for or enter a WSDL file (*screenshot below*). Select the WSDL file that defines the web service operation you want to request, and click **OK**.



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**.



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.



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.

8.5 Root Nodes

Each page data 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 **Project | List Usages of All Page Source Variables**.

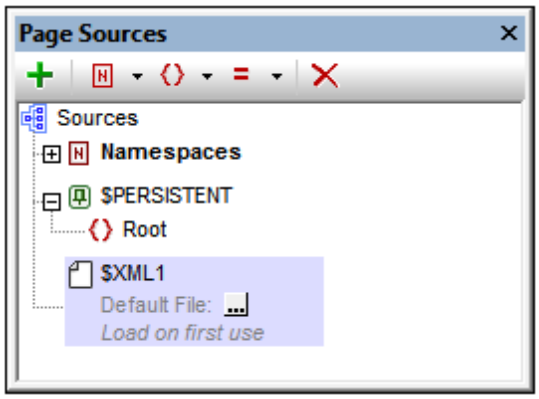
/ = $RootNodeName	• The **root node** is the topmost node of a tree. • A root node is represented by a variable of the form **$RootNodeName**, where RootNodeName is a name that identifies a particular tree's root node. An example is **$XML**. *See Names of Root Nodes below.*
<RootElement> <Element-1/> ... <Element-N/> </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**.) • 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 and repeating table 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 data sources are added. 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. • 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, or by importing an XML structure via its context menu's **Import** command. • 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). • $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 data sources.
$XML	XML documents, created manually or imported. Default data file is optional.
$HTML	HTML documents, created manually or imported. Default data file is optional.
$HTTP	Documents accessed via HTTP/FTP, REST, or SOAP. The requested resource provides the data.
$DB	The selected database table provides both structure and data.
$XQ	XQuery documents.
$FLOWFORCE	FlowForce jobs.
$MT_CONTACTS	Added when a Read Contacts action is added. Filled with data from the client's address book.
$MT_CALENDAR	Added when an Access Calendar action is added. Filled with data from

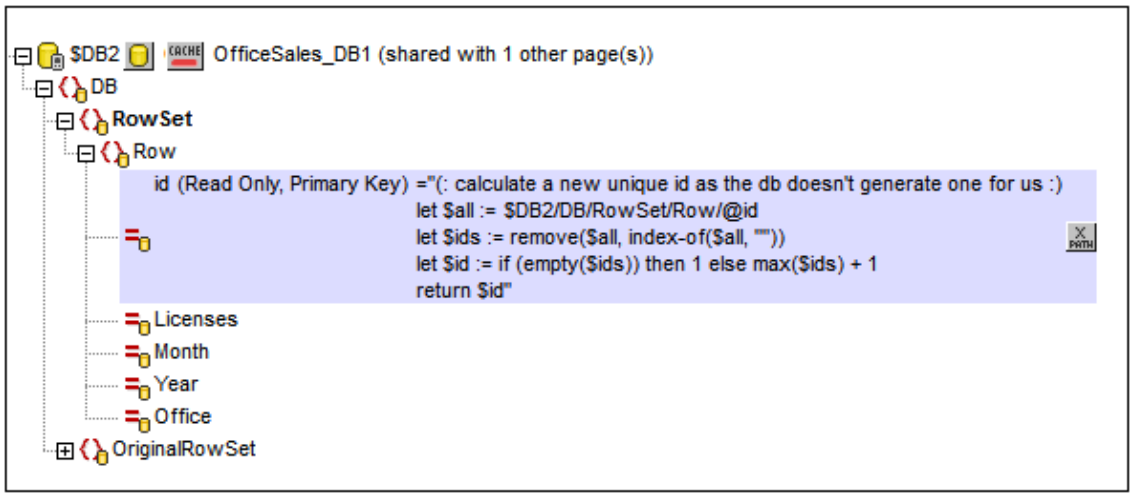
	the client's calendars.
$MT_EMBEDDEDMESSAGE	Added to the design when the **OnEmbeddedMessage** event is activated, or when an Embedded Message Back action is added
$MT_FILEINFO	Added when a Get File Info or Read Folder action is added. Holds information about the file/s and/or folder/s that are specified in these actions.
$MT_GEOLOCATION	Added when the Start/Stop Geo Tracking action or the Read Geo Data action is added to design.
$MT_NFC	Added when NFC Start/Stop 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.*
$MT_SERVICE	Created when a service design is created. It holds run time data about service triggers.

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.

8.6 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.



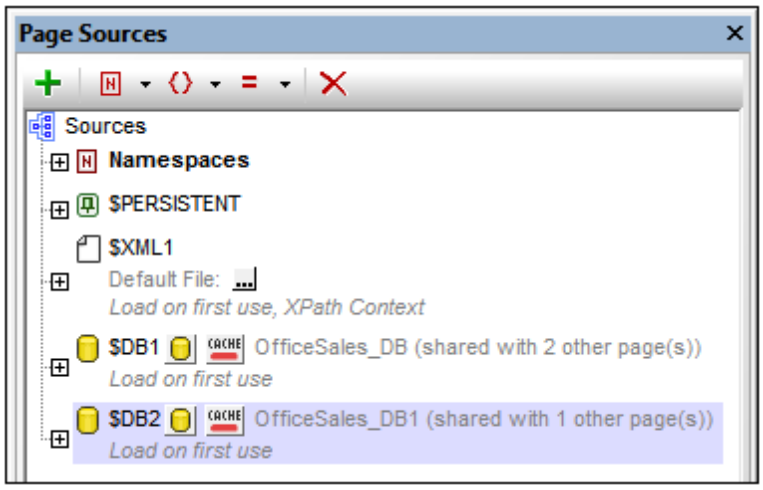
The structure of a page source is created in the following ways:

- Imported from an XML or HTML document, or an XQuery statement, when a page source is added (*see Adding Page Data Sources*), or subsequently
- Created manually in the Page Sources Pane by adding elements and attributes to a new empty XML tree (via the pane's toolbar icons)

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**.



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 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 data 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).

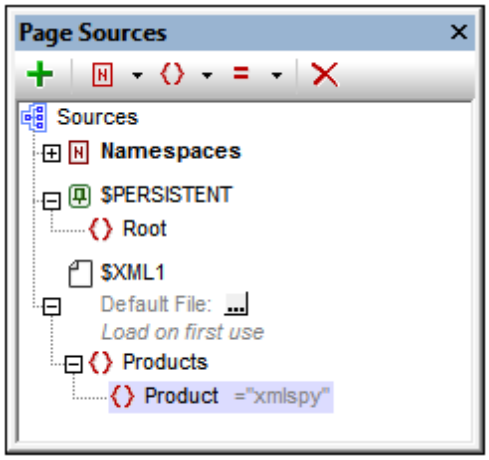
Page source links

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 onto the control.

The page source link of a control is **displayed in bold** in the data 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.

Tree Structure

When a page source is added to a page, a root node is created for that page source in the Page Sources Pane (*screenshot below*). Depending upon the type of the page source 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, 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

These methods are described below.

Importing an XML structure

Right-click the root node in the Page Sources Pane, 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).

Manually creating a tree structure

Elements and attributes can be added relative to any node in a tree structure (including the root node), and they can be deleted. Select a node in a data 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. A root node is created for the data source that is added. Only one child element can be added to a root node.
	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** commands.

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. Whether a page source is editable or read-only is defined at the time the page source is added. 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 data 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 data sources, the content of associated nodes is used for display purposes only. These associated nodes are know as **page source links**. A source node link is added to a control by dragging the tree node onto the control.

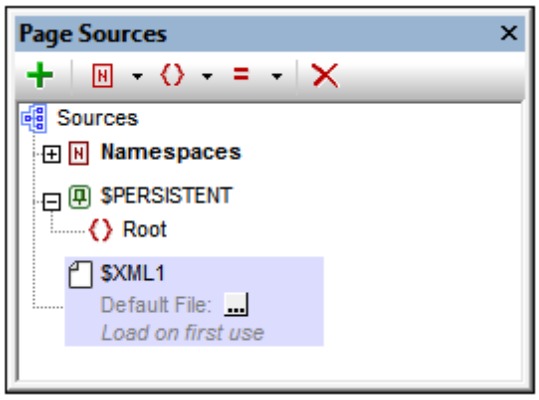
Assigning data to data sources

Data is assigned to nodes (in both editable and read-only data-source trees) in the following ways:

- *Assign a default file:* 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:* 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), the manually added node content overrides the previously assigned content.

Assigning a default file

An XML data 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*).



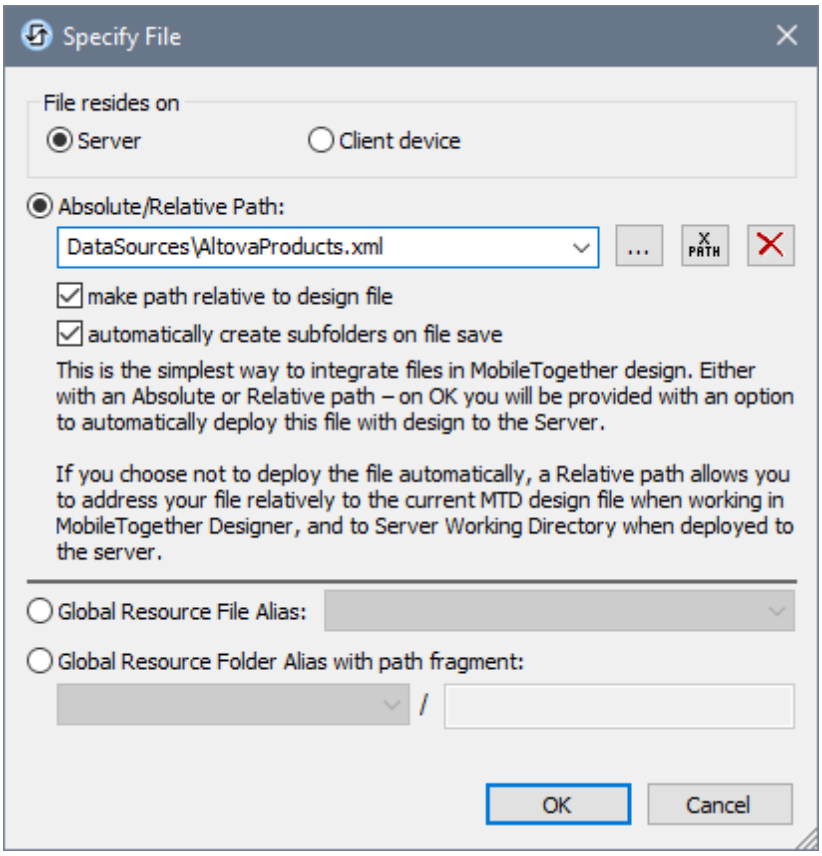
Click the **Additional Dialog** button to display the Specify File dialog (*screenshot below*), select the required file, and click **OK**. The assignment is made and the file path appears in the *Default*

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**. You can also manually create the structure of the data source 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.



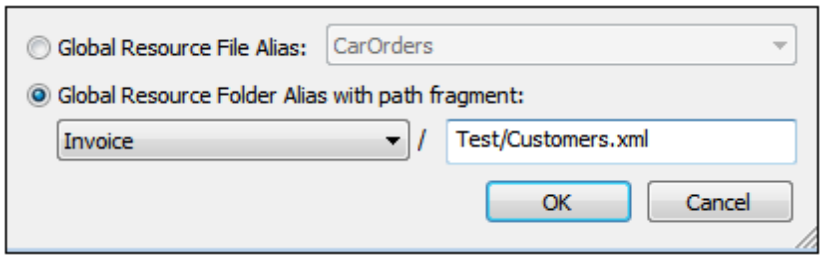
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design 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 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 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 for details. 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 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.

- *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**). See the section Altova Global Resources for details.

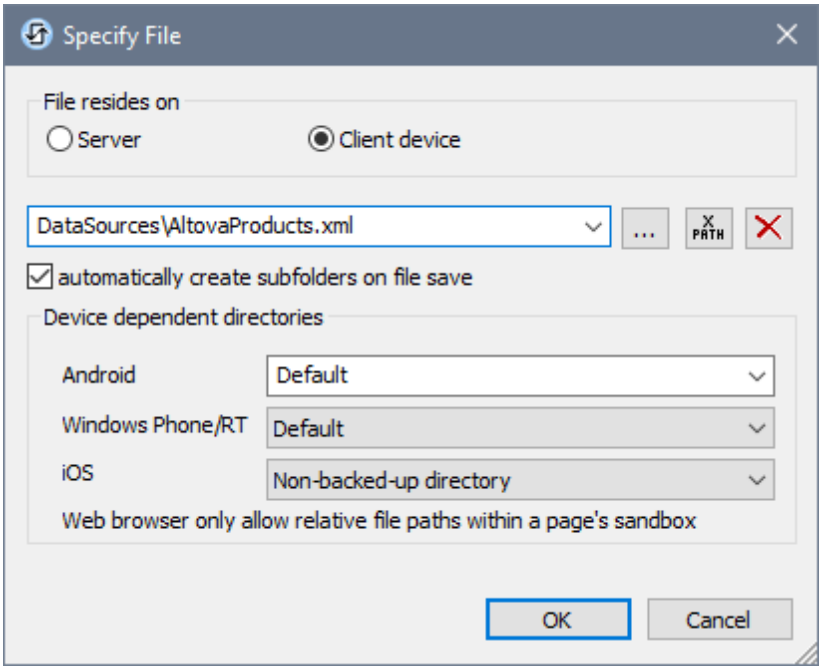
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



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**). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 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.



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 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 (*Source File* and *Target File* options), Load/Save File, Load/Save Image, and Load/Save Binary File.

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 (**Tools | Options**).

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** commands.

8.7 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 (*screenshot below*) contains all the namespaces that have been declared for the project, irrespective of what page is currently active in Page Design View.

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 (*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 (*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.

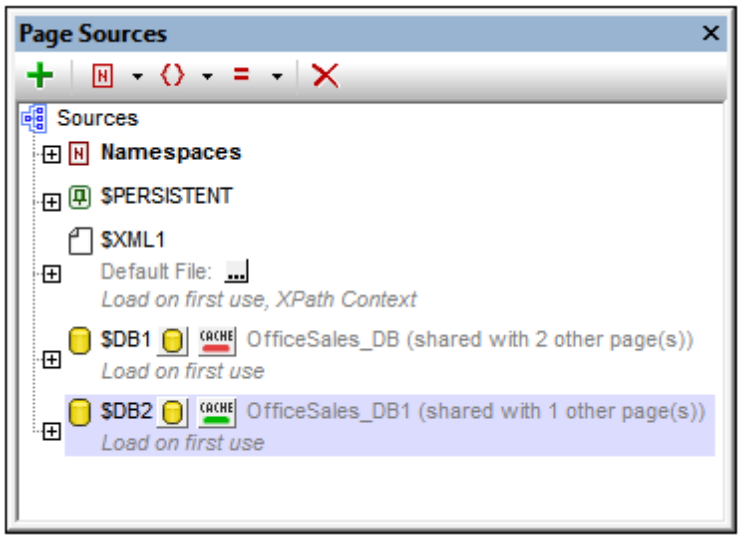
To delete a namespace, select it and click **Delete** in the pane's toolbar.

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.

8.8 Caches

The data in external resources (XML files and DBs) that are linked to a page source can be cached on the server. Whether a page source has been cached on the server is indicated by the color of the Cache icon of the data source (*see screenshot below*). A red cache symbol indicates that the page source has not been cached. A green symbol indicates that a cache has been created for that page source. If a page source is **not** linked to an external resource (for example via a default file), then it has no cache symbol (as for `$XML1` in the screenshot below).



Reasons for caching

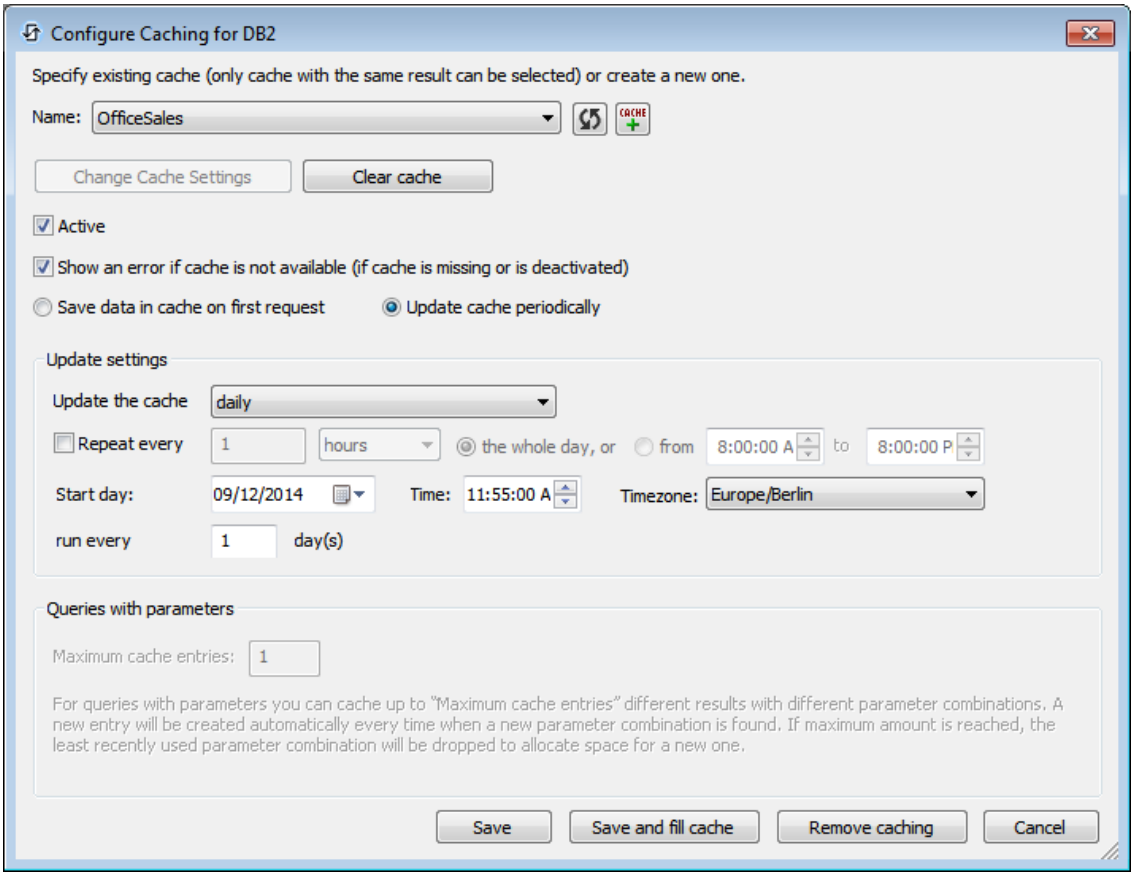
There are two main reasons to create caches: (i) If a page data source generates reports slowly (for example, a large database); (ii) If a page source is not modified often. In cases of both these types, execution of a solution would be faster if data is taken from data caches on the server.

All caches that have been saved to the server can be viewed in the *Cache* tab of the MobileTogether Server configuration page (web UI).

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 page sources of different designs, providing that the underlying data structure is compatible. If a page 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.

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*).

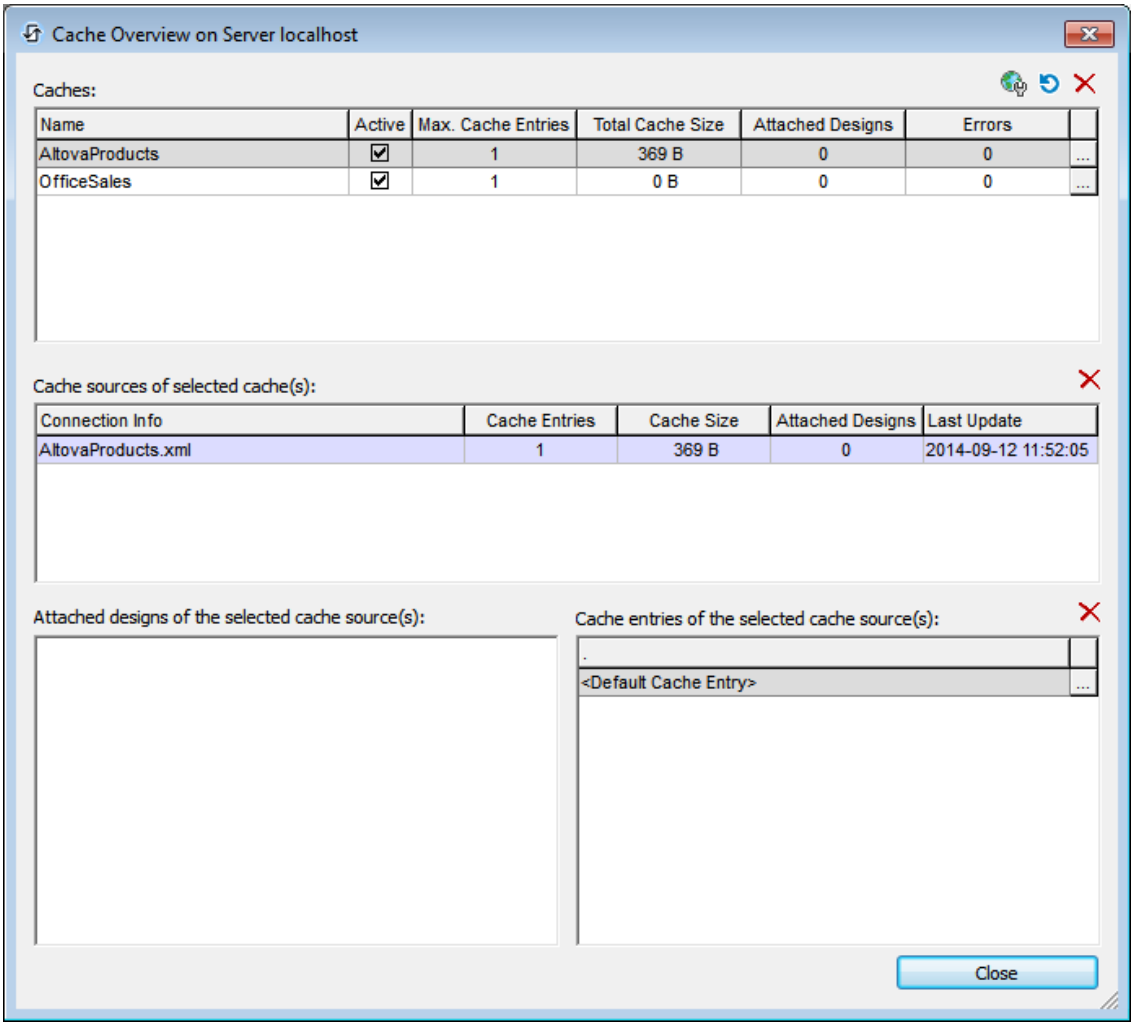


In this dialog, you can do the following:

- Refresh the connection to the server to check for the latest caches that match the structure of this page source.
- Add a new cache for this page source.
- Click **Change Cache Settings** to edit the settings of a cache that already exists.
- Activate/deactivate the cache.
- Specify whether an error should be displayed if the cache is missing when the solution is run.
- Specify the frequency with which the cache should be filled.

Cache Overview

The Cache Overview dialog (*screenshot below*) is accessed with the menu command **Project | Cache Overview**.



The dialog provides an overview of all the caches on the server. In it, you can also do the following:

- Activate/deactivate caches.
- Delete caches.

8.9 Context Menus

Commands in the context menus of items in the Page Sources Pane are described below. They are organized into two groups:

- Context menus of root nodes
- Context menus of tree nodes.

Context menus of root nodes

The commands listed below are available in the context menus of root nodes (`$XML`, `$DB`, `$HTML`, etc). In addition to the commands that are common across all types of data sources (XML, DB, HTML, etc), some types of data 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.

▼ 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 exactly which data to transmit to 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 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 toggle command specifies whether data in the tree is stored on the server only, the client only, or shared by both. Note that if it 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

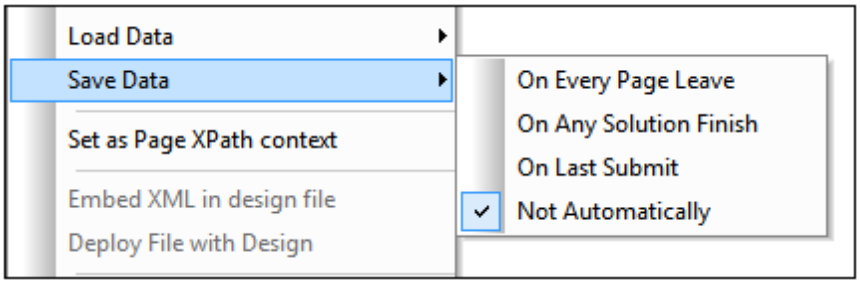
This command rolls out a sub-menu with the following mutually exclusive options (only one can be selected):

- *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, Load from File, or Load from HTTP actions to load it. Alternatively, it can be created completely from scratch with the Append Node and Insert Node 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, Save to File, or Save to HTTP/FTP actions to save data.

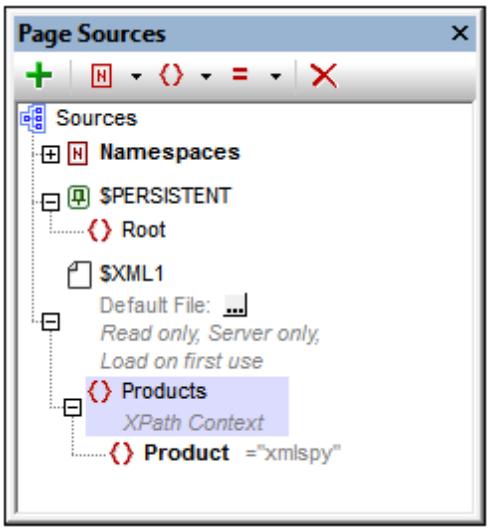
The default is *Not Automatically*.

▼ Reset data

This command enables the data of the selected root node (page source) to be reset on every page leave. Two options are possible: (i) the page source data is reset when the page is left; (ii) teh page source data is not automatically reset when the page is left. (In the second case, if a data reset is required only under other conditions than a page-leave, then use the Reset action.)

▼ 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.



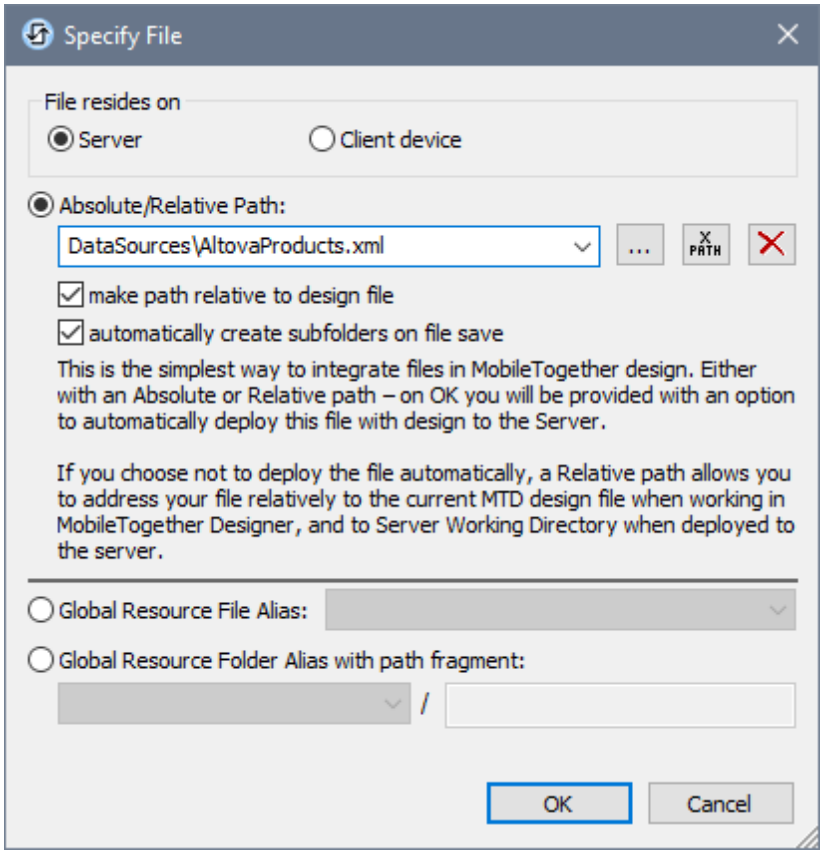
This command can be toggled on and off. So, you can enable a node as the XPath context node of the page, or you can toggle off a node's setting as the XPath context node of the page. If a node is set as the XPath context node when another node already has this setting, then the setting is toggled off 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 data source. However, in order for the data to be used, the default file must have the same structure as the data source. Note that, when a default file is assigned to a data 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 data source 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.



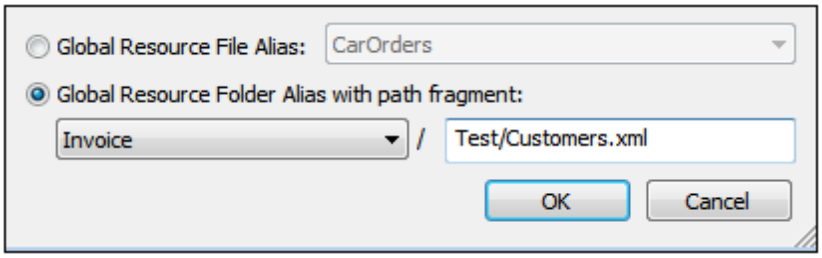
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design 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 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 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 for details. 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 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.

- *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**). See the section Altova Global Resources for details.

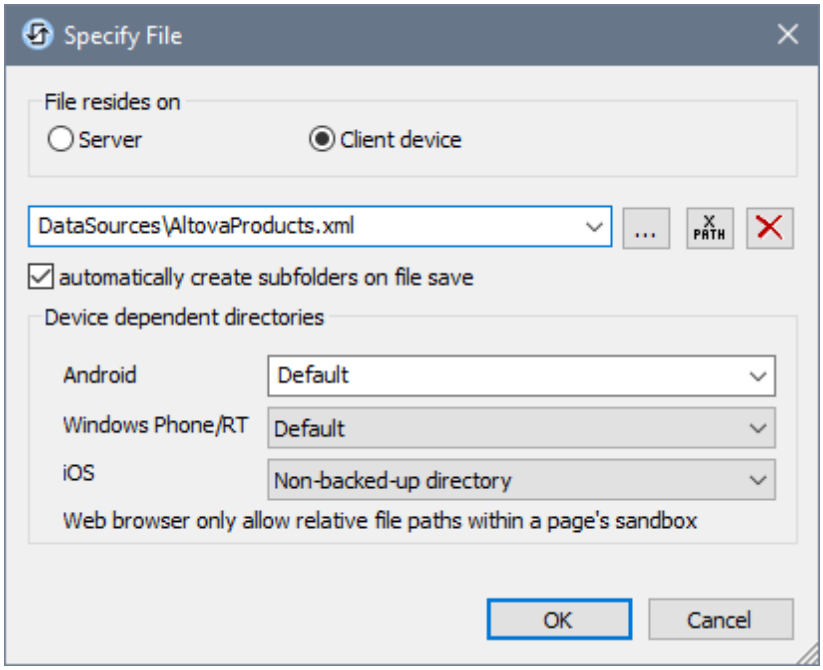
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



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**). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 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.



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 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 (*Source File* and *Target File* options), Load/Save File, Load/Save Image, and Load/Save Binary File.

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 (**Tools | Options**).

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.

▼ Embed XML in design file

This command is enabled when the root node of an XML page source is selected that has a default file assigned to it. 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 and Embed XML in Design File 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.

- Toggling the command on selects the check box of the file in the Files Pane, thus deploying it.
- Toggling the command off de-selects the file in the Files Pane; 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.

▼ List variable usage

The root node of every page source is a variable, for example **$XML1** or **$DB1**. The **List Variable Usage** command lists, in the Messages Pane, 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, then the command is enabled.

▼ 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 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.

▼ 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.

Note: When a structure is imported from a JSON file, the file is set as the default file 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.

▼ 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 data 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 data 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 data 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.

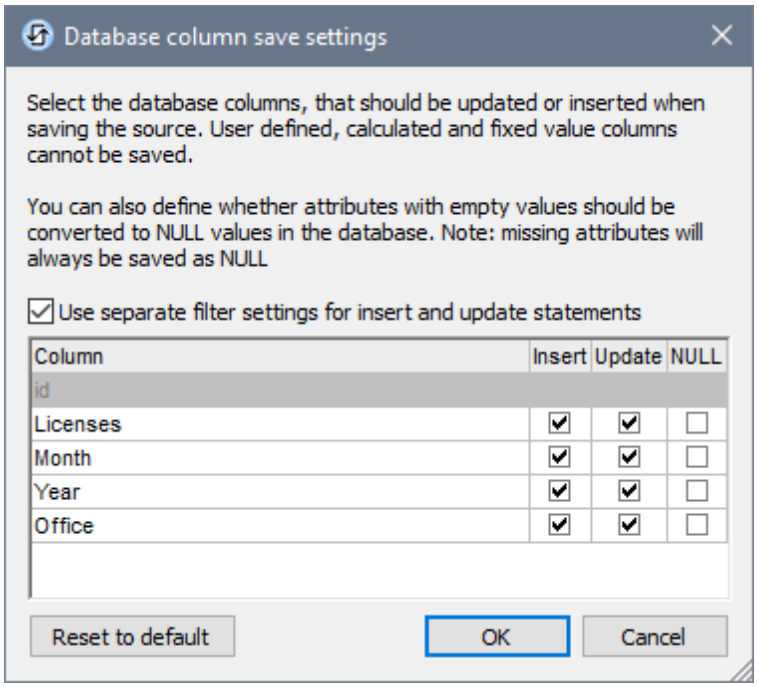
To create an `OriginalRowSet` for a page source, right-click the root node of the page source and toggle on the command **Create OriginalRowSet**.

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. User modified data is saved in the main structure

created from the data source. When modified data is saved back to the DB, the `OriginalRowSet` structure is modified to contain the data newly saved to the DB. 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.

▼ 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.



The dialog displays the columns of the DB data 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. If you wish to reset the *Save settings* so that all columns are updated, click **Reset to default**.

- HTTP/FTP Request Settings

  This command is enabled for root nodes of the HTTP/FTP type (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, RESTful API Request Settings, Choose WSDL File.

- 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 | Caches.

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 data 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

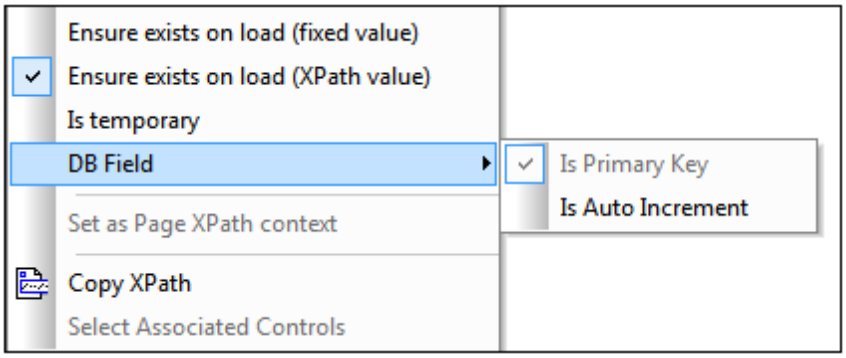
provided for the selected node when the page is loaded. On clicking the command, the Edit XPath/XQuery Expression Dialog is displayed. Enter the XPath expression to generate the value of the node. This command is a toggle command. So, clicking it when an XPath value is already assigned will remove the 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.

- DB field

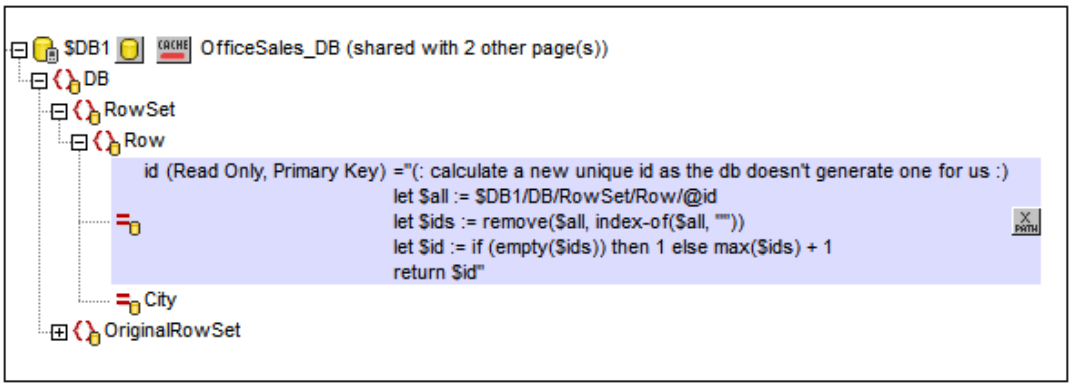
  This context menu item is available for DB nodes, and has a sub-menu with two commands:



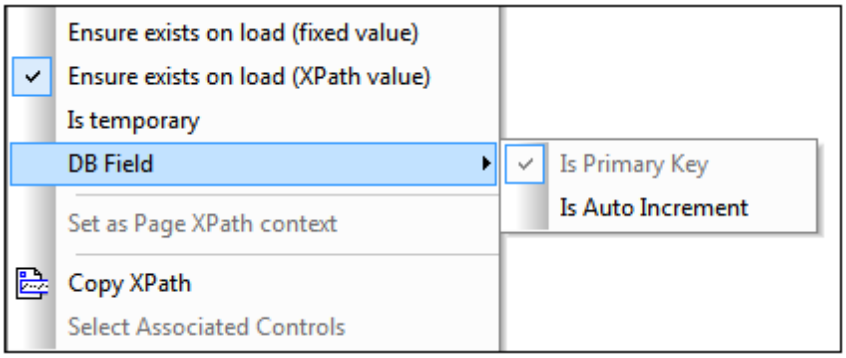
  - *Is Primary Key:* Sets the selected node as the primary key column if a primary key has not already been auto-detected during retrieval from the DB.
  - *Is Auto Increment:* Sets the selected node to auto increment. The node then becomes read-only.

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*).

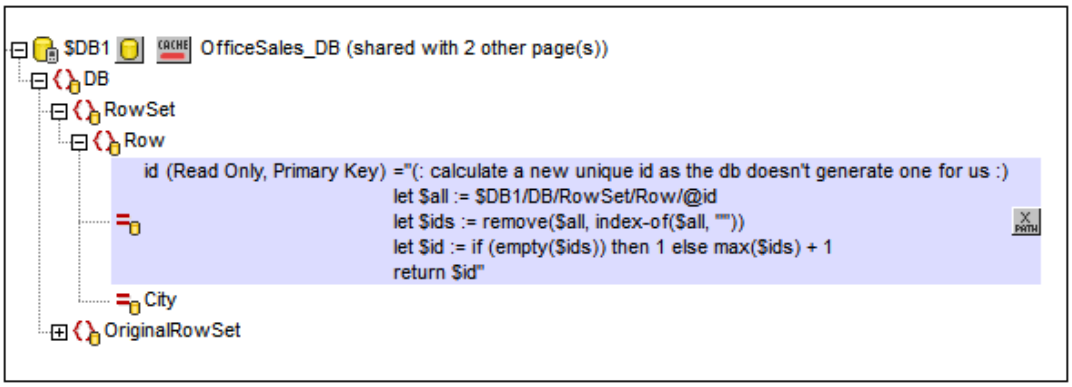


If auto-retrieval of this information was not successful, the context menu of tree nodes contains toggle commands that enable you to correctly annotate nodes (*see screenshot below*).



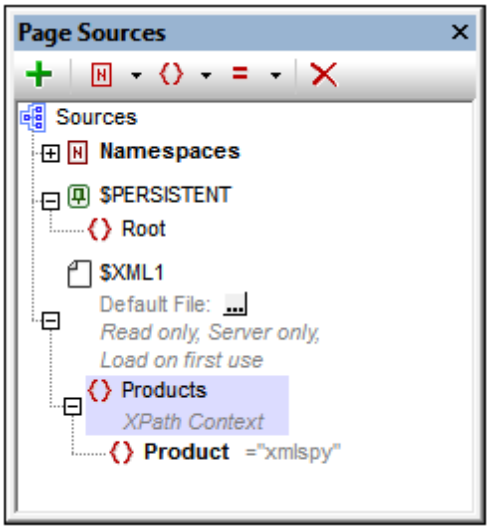
If the primary key column is not auto-incrementing, new primary key values for appended rows must be automatically generated using an XQuery expression. This is because primary key columns cannot be edited. The XQuery expression is inserted by using the primary-key node's context menu command, **Ensure 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
```



- 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.

- 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.

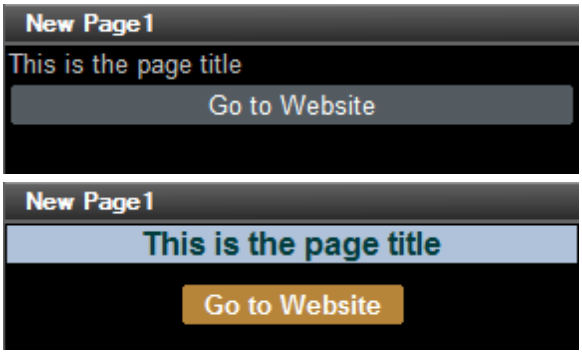
Chapter 9

Controls and Control Events

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 that displays the page title, and (ii) a button control 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 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. To add a control to the design, drag it from the Controls Pane and drop it at the required location in the page design. You can then do the following:

- *Link the control to a data source* (available in the Page Sources Pane) so that data from a node in the data 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) 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 to execute when a control-related event is triggered (for example, when a button is clicked). Define control actions by right-clicking the control and selecting **Control Actions**. The available actions are described in the section Actions.
- *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. The properties of controls are described in the section Controls. 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 describes each control separately: its function, its properties, and its event/s (for example, the button control has one event, the `OnButtonClicked` event)
- Control Events 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. After a control has been placed in the design, you can do the following:

- *Controls can be linked to data sources* (available in the Page Sources Pane) so that data from these sources can be displayed in the design and processed. Typically, an association between a control and a data source node is defined by dragging a data node from its source tree (in the Page Sources Pane) onto the control. A data association of this type is called the **page source link** of the control.
- *Set control actions*: Control actions determine the functionality to be executed when a control-related event is triggered (for example, when a button is clicked). You define control actions by right-clicking a control, and selecting **Control Actions**. Control actions are described in the section Actions.
- *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. Each control's properties are described in the sub-section of this section.

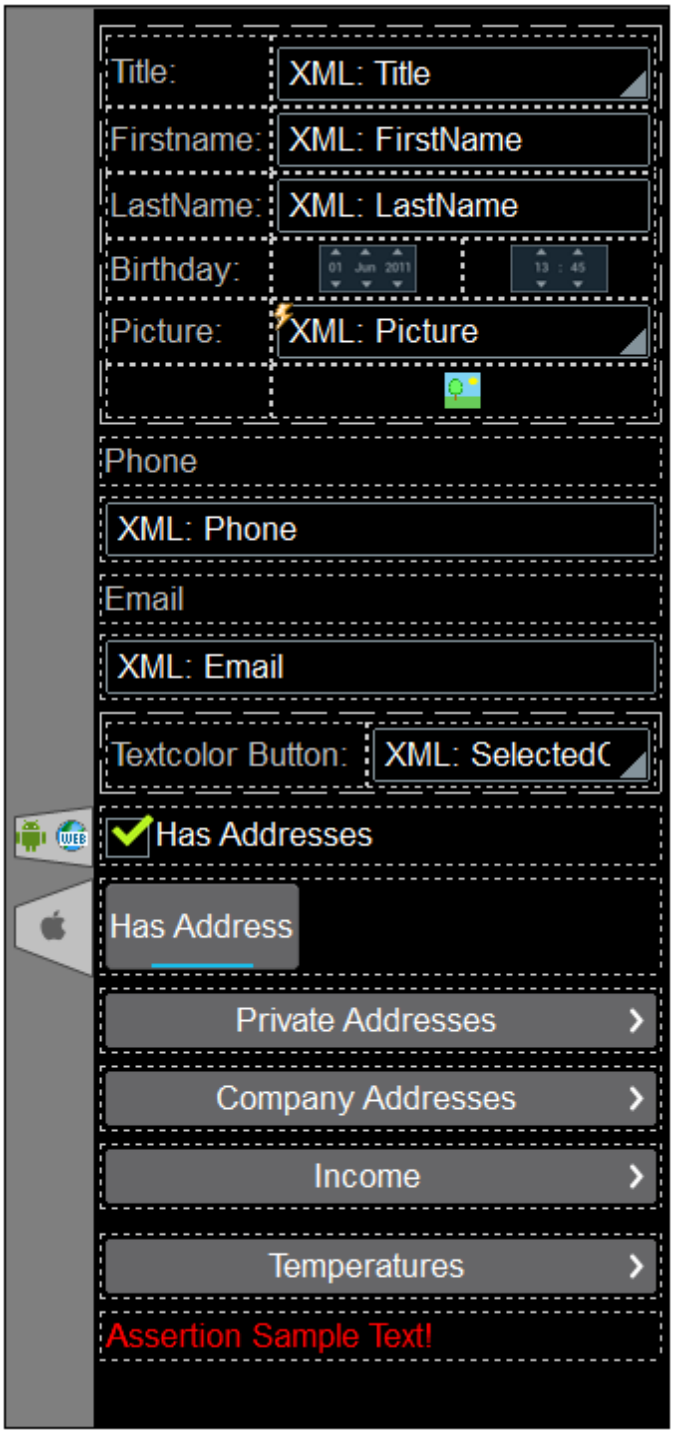
Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

List of controls

Given below is a list of available controls, arranged alphabetically.

- Assertion Message
- Button
- Chart
- Checkbox
- Combo Box
- Date
- DateTime (iOS)
- Edit Field
- Horizontal Line
- Horizontal Slider
- Image
- Label
- Radio Button
- Rich Text
- Signature Field
- Space
- Switch
- Table
- Time
- Vertical Line
- Video



Context menu

Each control in the page design has a context menu. The following control-related commands are common to most context menus.

- Delete Page Source Link
  - *Description*

    This command is enabled for those controls that can be associated with a data source node and for which an association exists. **Delete Page Source Link** deletes the association between control and data 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, 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.

- Enter Text
  - *Description*

    Enabled for controls in which text can be entered. Rolls out a sub-menu with the following options:

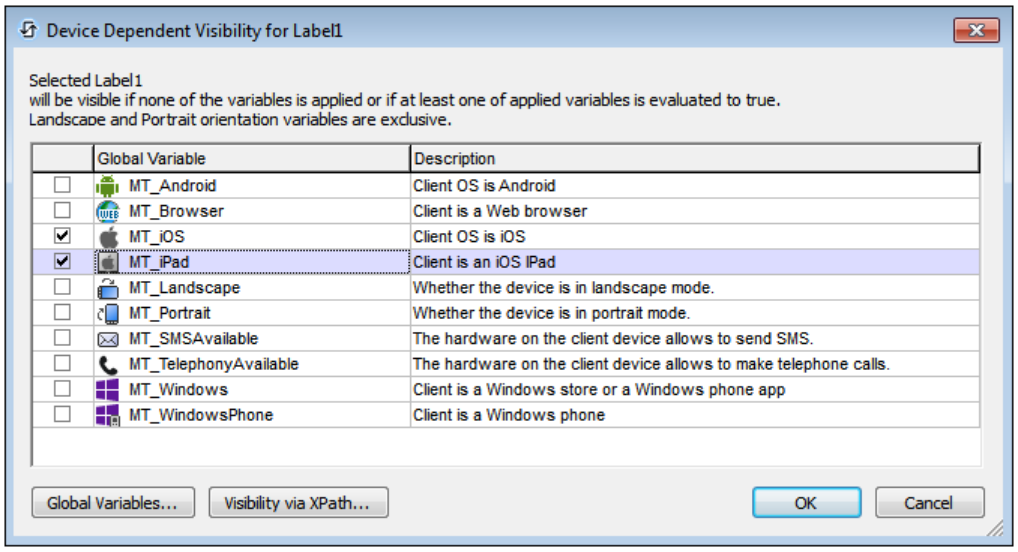
    - *Directly:* To enter static text directly as the text of the control
    - *XPath:* Displays the Edit XPath/XQuery Expression dialog, in which you can enter the XPath expression that selects the text of the control
    - *XML Node:* Refers to the option of displaying the content of an XML node as the control's text. Clicking the option displays a hint that a data source node can be dragged from the Page Sources Pane 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*

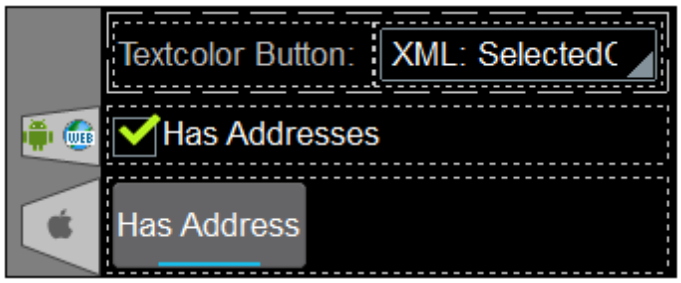
    Displays the Localization dialog, in which you can define the localization (translation) of strings that appear in various controls of the project. This command has the same effect as the **Project | Localization** command. For a description of localization, go to the description of the **Project | Localization** command.

- 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**.



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, in which you can set actions for various page events. This command has the same effect as the **Page | Page Actions** 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** command.

- Actions Overview
  - *Description*

Displays the Actions Overview dialog of the currently active page. This command has the same effect as the **Page | Actions Overview** command. For more information, go to the description of the **Page | Actions Overview** command.

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 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 control is inserted.

Notes

- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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, 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.

▼ Multiline

Sets multiline input/display on or off (`true/false`). The default is `false`.

▼ 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-in-words. 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`** 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'`**.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate

space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Text Size Auto-Fit

  Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

  If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

  The auto-fit function cannot be used if the **Multiline** property has been set to **true**.

  Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
  - 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 disabled, use the **`Background Color (Disabled)`** property.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate

space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Margin Bottom

  Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Style Sheet

  The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 data source node (by dragging the node onto the button). Alternatively, you can also select an icon from the options available in the `Button Look` property combo box. When an icon is selected, the `Text` property and all text-formatting properties are no longer available. To make them available, select the no-icon option of the `Button Look` property. 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, in which you can specify the required action.

Notes

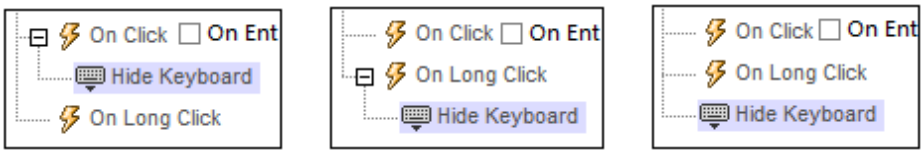
- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the data source tree are displayed in a bold font. Tree nodes that are not page source links are displayed 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane 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.

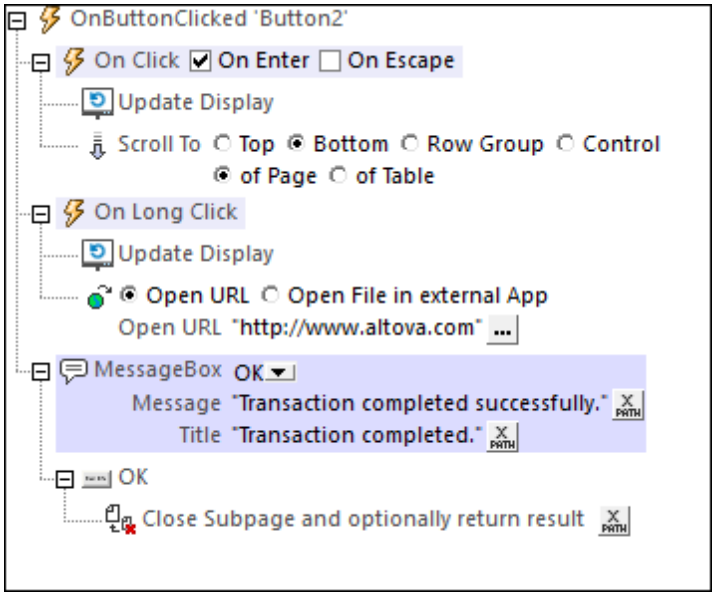
OnButtonClicked (OnClick, OnLongClick)

The end user can click the control in one of two ways: a short tap or a longer press. A sequence of different actions can be specified for each type of click. An additional sequence of actions can be performed after those of the end-user click. The additional sequence is defined after the `On Long Click` event.



- **`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 for the different click events, for example, as shown in the screenshot below.



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` and/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 gets the focus and is "clicked" when the respective key (**Enter** or **Escape**) is tapped.

This setting can also be accessed via the control's `On Enter/Escape` property. For more details, see the description of the property below.

Note: These shortcut assignments can be viewed by selecting the **Page | Show/Define Tab Order** menu command.

Note: This feature is available only on Web clients and Windows clients.

Button properties

The control's properties are available in the Styles & Properties Pane, 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.

- 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 data 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. For some controls you can choose between the two methods of entering the property value: right-click the property and select the entry method you want from the context menu (fixed-value or XPath). For other controls, there is only one method of entering the property value.

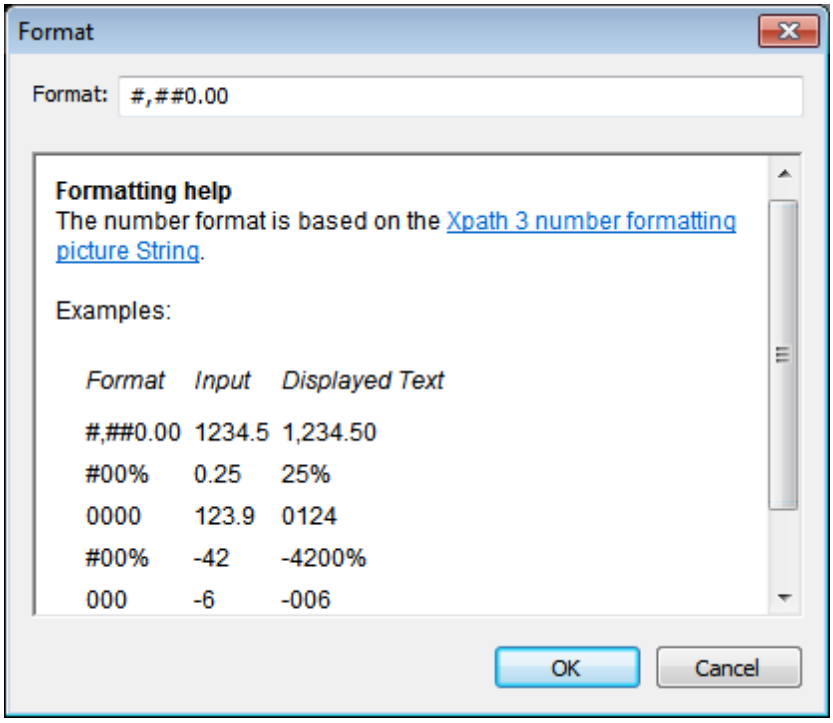
  Note: The **`$MTControlValue`** variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

- Multiline

  Sets multiline input/display on or off (`true/false`). The default is `false`.

- Number Format String

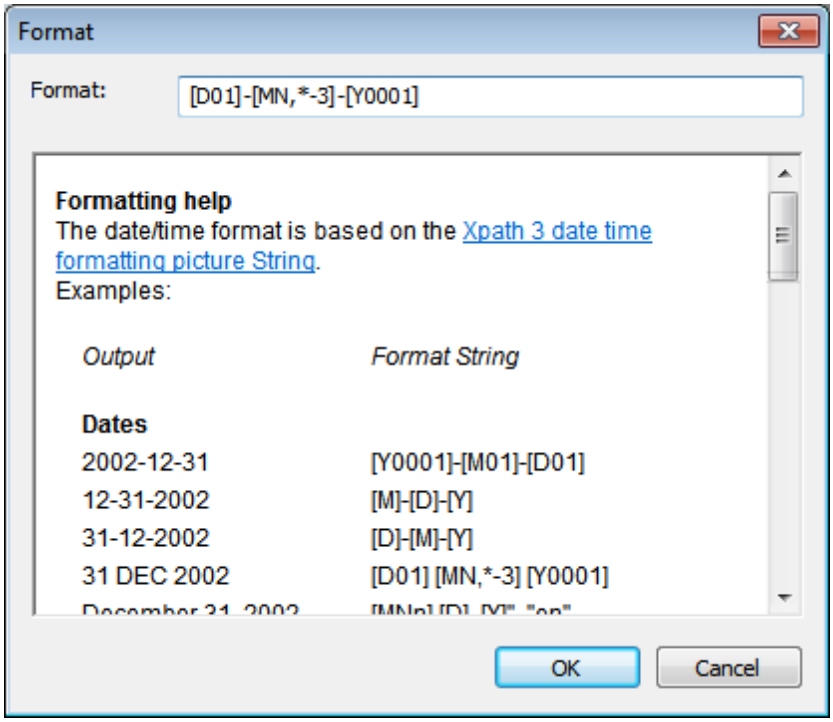
  Click the **Additional Dialog** button and enter a number format in the Format dialog that appears (*screenshot below*).



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*).



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 Look

Adds a predefined icon (instead of text) as the button display. Add an icon by selecting one from the dropdown list of the property's combo box. Available icons include: *Plus, Minus, Share, Email, Print, Delete, Edit*, *Import, Export, Play, Play Reverse, Pause, Resume, Stop, Microphone, Calendar, Time, Fast Forward, Fast Rewind* and *Search*, with most icons available on a transparent or opaque button background. The default value of the `Button Look` property is no icon. When an icon is added as the display of the button, any previously entered button text is removed. Additionally, all the control's text-formatting properties are removed. To switch back to a button-text display (instead of an icon), select the no-icon option, and enter the button text, either as the value of the `Text` property, or by double-clicking the button and directly entering the button text there.

Control Action

Click the **Additional Dialog** button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** 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-in-words. 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`** 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'`**.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Text Size Auto-Fit

  Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

  If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

  The auto-fit function cannot be used if the **`Multiline`** property has been set to **`true`**.

  Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
  - 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 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

▫ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the

resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

Maximum Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The Maximum 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 value*: select a pixel 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. The default is wrap_content for all controls.

Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic

variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the

resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($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** 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 this control gets the focus and is "clicked" when the **Enter** key is tapped.
- `OnEscape`: Specifies that this control gets the focus and is "clicked" when the **Escape** key is tapped.
- None: No action when any key 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 Control Events).

Note: This feature is available only on Web clients and Windows clients.

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

Chart

The Chart control enables data from a source data file to be displayed in the form of a chart. The available chart types 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. 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 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

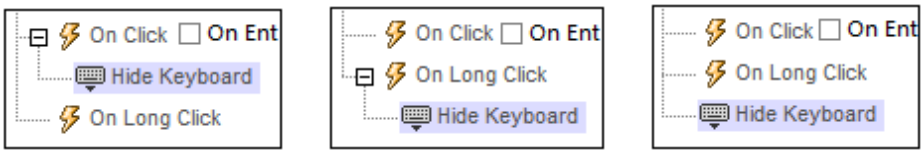
Chart events

The **`OnChartClicked`** event is available. To define actions for the chart's **`OnChartClicked`** event, right-click the chart and, from the context menu that appears, select **Control Actions for OnChartClicked**. 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.

OnChartClicked (OnClick, OnLongClick)

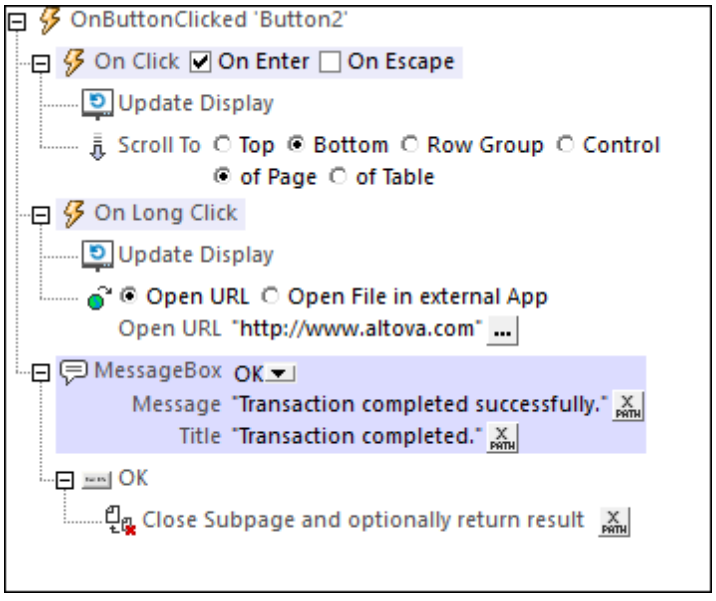
The end user can click the control in one of two ways: a short tap or a longer press. A

sequence of different actions can be specified for each type of click. An additional sequence of actions can be performed after those of the end-user click. The additional sequence is defined after the `On Long Click` event.



- **`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 for the different click events, for example, as shown in the screenshot below.



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` and/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 gets the focus and is "clicked" when the respective key (**Enter** or **Escape**) is tapped.

This setting can also be accessed via the control's `On Enter/Escape` property. For more details, see the description of the property below.

Note: These shortcut assignments can be viewed by selecting the **Page | Show/Define Tab Order** menu command.

Note: This feature is available only on Web clients and Windows clients.

Chart properties

The control's properties are available in the Styles & Properties Pane, 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.

- 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.

- ID

  This property must be entered when the chart is placed in a repeating table or repeating row of a dynamic table. 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, of the chart to be generated.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($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, of the chart to be generated.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the

current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

Maximum Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The Maximum 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 value*: select a pixel 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 control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 the image width would be

`concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the `$MT_CanvasX` and `$MT_CanvasY` variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values

returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

On Enter/Escape

Takes one of three values:

- OnEnter: Specifies that this control gets the focus and is "clicked" when the **Enter** key is tapped.
- OnEscape: Specifies that this control gets the focus and is "clicked" when the **Escape** key is tapped.
- None: No action when any key 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 Control Events).

Note: This feature is available only on Web clients and Windows clients.

Style Sheet

The Style Sheet property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 an XML data 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, 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Check Box events

The **`OnFinishEditing` event** is available. For a description of the actions that can be defined for this event, see the Actions section.

Check box properties

The control's properties are available in the Styles & Properties Pane, 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.

- 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 data 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. For some controls you can choose between the two methods of entering the property value: right-click the property and select the entry method you want from the context menu (fixed-value or XPath). For other controls, there is only one method of entering the property value.

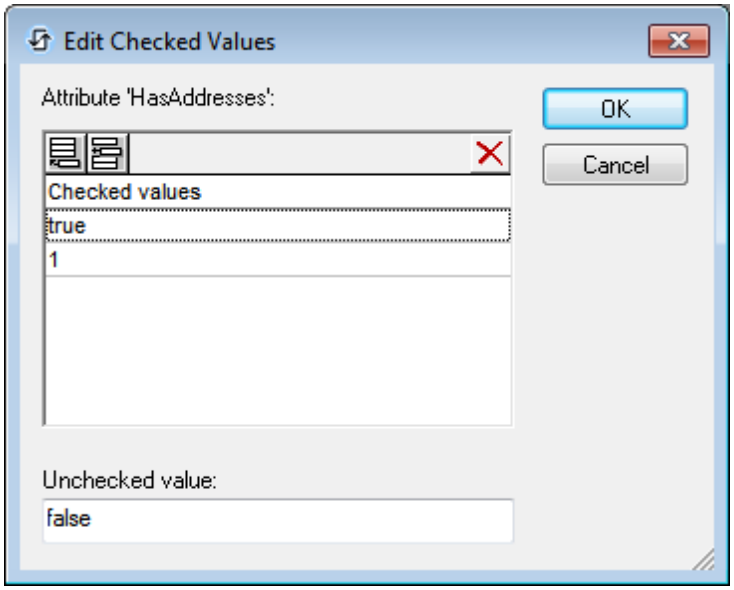
  Note: The **`$MTControlValue`** variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

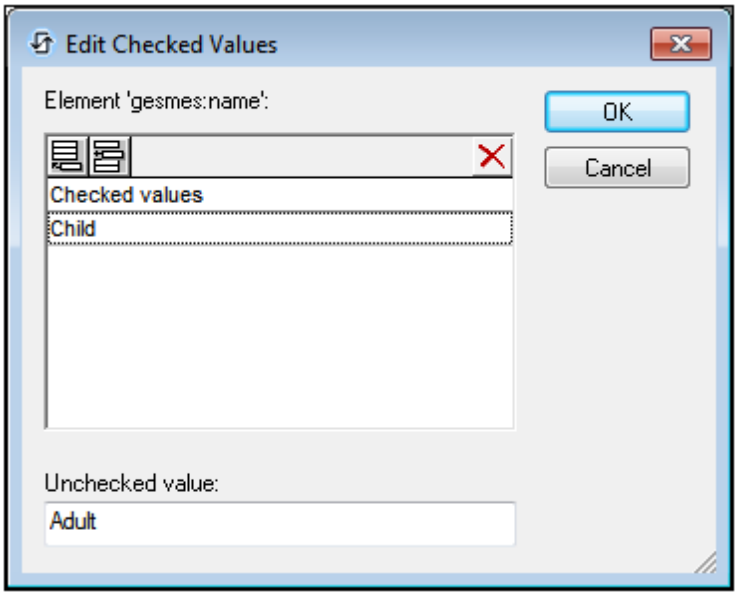
- Multiline

  Sets multiline input/display on or off (`true/false`). The default is `false`.

- Checked Values (true/false)

  Provides an XML data value for the selected/unselected state of the control. Click the **Additional Dialog** button to display the Edit Checked Values dialog (*screenshots below*). Enter a value for the checked (selected) and unchecked (unselected) state. The value corresponding to the control state (selected/unselected) chosen by the end-user will be entered as data in the XML node associated with the control. To specify which node will receive the value, make a page source link from the check box to an XML data source node (by dragging the node on to the check box).





- 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`** 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 at

least one XML value. These XML values are defined in the `Checked Values` property.

The `Auto Correct Value` property has two possible values: `true` or `false`. If the property is set to `true`, XML values are automatically corrected to the values defined for the checked and unchecked states in the `Checked Values` property. For example, if the checked value has an XML value of `child` and the unchecked value has an XML value of `adult`, then, if the control is checked, the XML value will be corrected to `child` in case something else is entered. If the control is unchecked, the XML value will be `adult`. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction. A correction would be necessary, for example, if the control is associated with a node having content that is not a legitimate XML value for the current state of the control. 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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** 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-in-words. 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`** 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'`**.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Text Size Auto-Fit

  Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

  If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

  The auto-fit function cannot be used if the **Multiline** property has been set to **true**.

  Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
- 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 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components

maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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** 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 control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 data source) are specified with the `Combo Box Entry Values` property (*see below for details*).
- *Associated XML node*. A node from the Page Sources Pane 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.

Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Combo Box events

The **`OnFinishEditing event`** is available. For a description of the actions that can be defined for this event, see the Actions section.

Combo box properties

The control's properties are available in the Styles & Properties Pane, and are listed below in the order in which they appear.

- Name

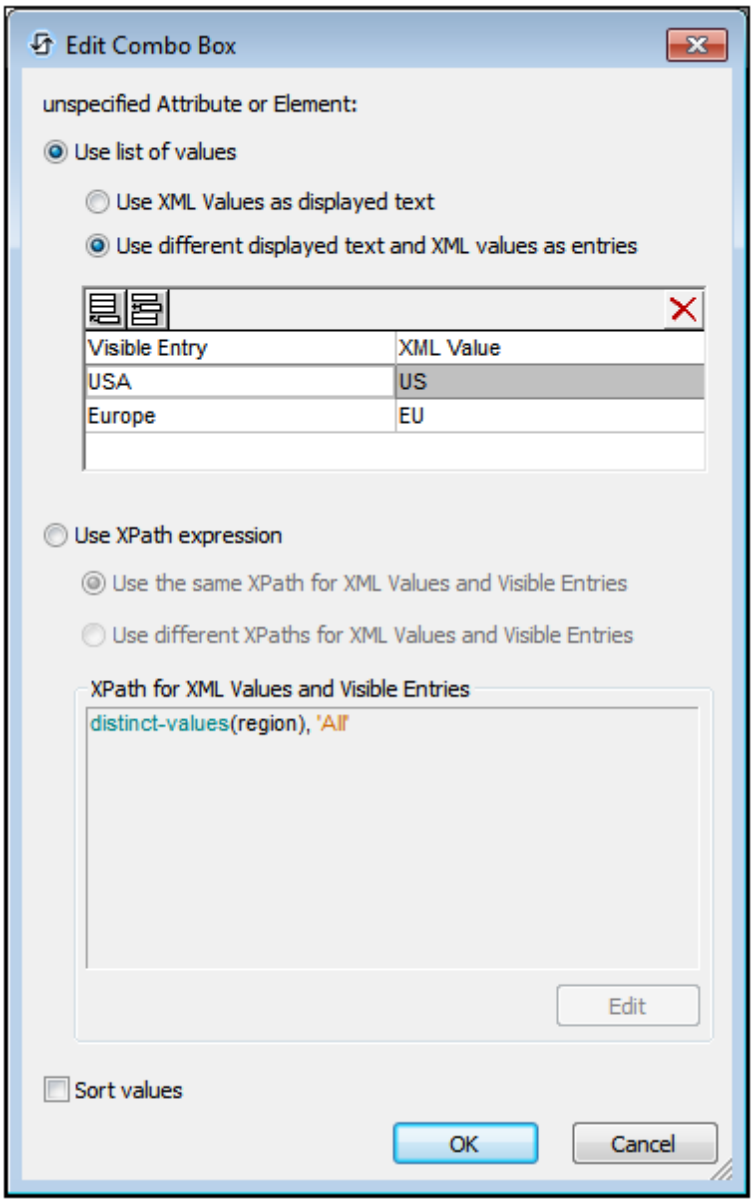
  The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

- Multiline

  Sets multiline input/display on or off (`true/false`). The default is `false`.

- 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.



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) list of values, or (ii) XPath expressions.
2. If you select *Use List of Values*, you can specify the text of entries that appear in the dropdown list of the combo box and the XML values that these entries map to. You can choose between using (i) the entry text as the XML value, and (ii) text-to-value mappings. You could also use an XPath expression to create the visible entries and XML values. The items in the sequence returned by an XPath expression are used for visible entries and/or XML values. You can specify: (i) that the same XPath expression be used for visible entries and for XML values, or (ii) that different XPath expressions be used for visible entries and for XML values. In the latter case,

a one-to-one index mapping between the items of the two sequences determines the correspondence of visible entry to XML value. If the number of items in the two sequences are not equal, an error is reported.

3. 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.
4. Click **OK** to finish.

Note

- Using an XPath expression to select the items of the combo box drop-down list enables you to create combo boxes with dynamic entries from an XML data source.
- If the items in the drop-down list of the combo box are obtained from an XML data source, 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`** 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`, the display items in the dropdown list items are automatically corrected so that only those items defined in the list are displayed. Any non-defined items will be removed. A non-defined item could enter the list, for example, via the node that is the page source link of the combo box. If the display values are auto-corrected, the items of the dropdown list will appear in the order in which they are defined. The property's default value is `false`.

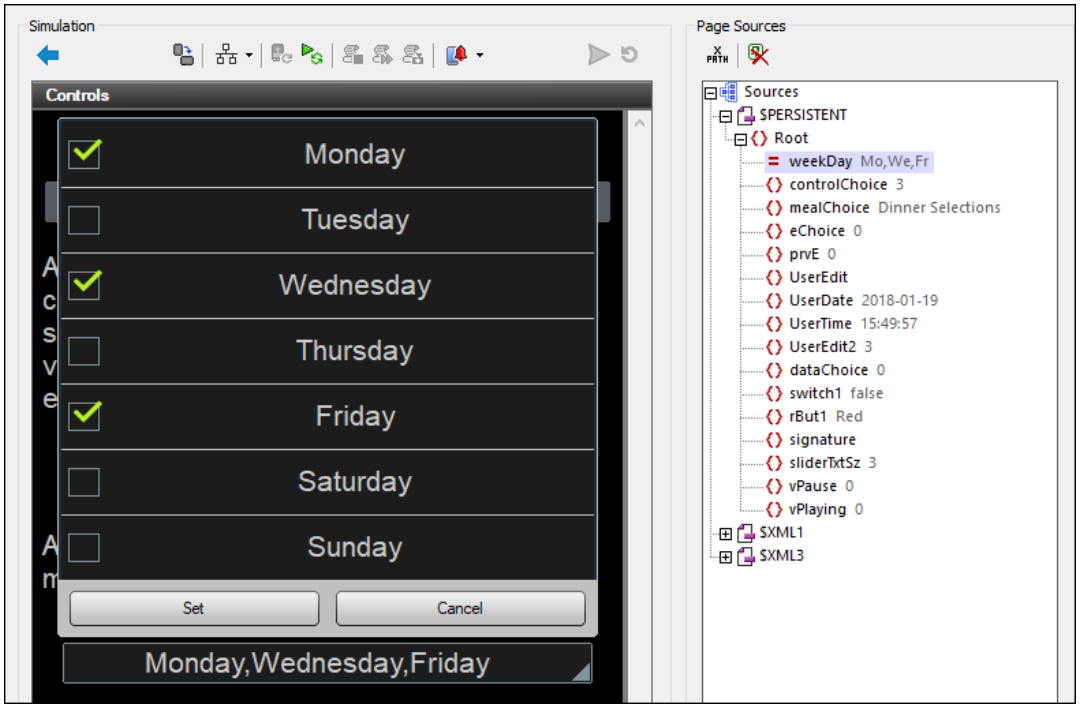
▼ Multi Select

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. 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. The separator would have been selected via the **`Multi Select Separator`** property (*see below*).



Note: If values in a node containing a multi-selection need to be read individually, the string must be tokenized using the same separator that was used when constructing the string.

Multi Select Separator

This property is enabled when the `Multi Select` property is set to `true`. It specifies the separator to be used in the string that is returned. See the the `Multi Select` property for details.

You can either select a separator from the available choices or enter a character you want to use as the separator.

Control Action

Click the **Additional Dialog** button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** 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-in-words. 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`** 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'`**.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Text Size Auto-Fit

Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

The auto-fit function cannot be used if the **`Multiline`** property has been set to **`true`**.

Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
  - 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 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to

wrap_content. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic

variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the

resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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** 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 control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

Date

Date controls are used to format dates obtained from a node in a data 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Date events

The **`OnFinishEditing event`** is available. For a description of the actions that can be defined for this event, see the Actions section.

Date properties

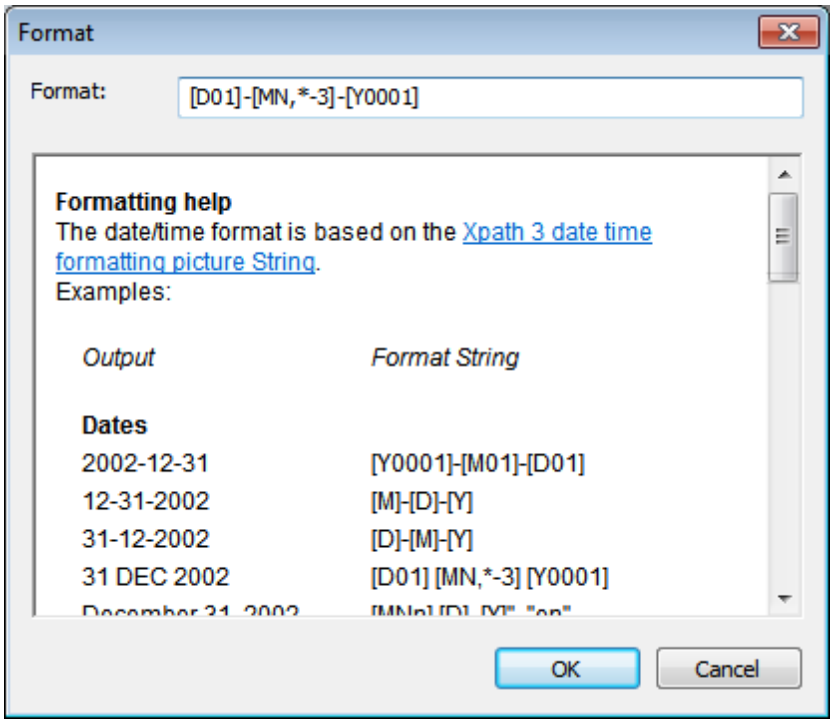
The control's properties are available in the Styles & Properties Pane, and are listed below in the order in which they appear.

Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

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*).



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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** 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 control. (If there are multiple Assertion Message 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 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 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 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-

in-words. 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** 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'**.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Text Size Auto-Fit

Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

The auto-fit function cannot be used if the **Multiline** property has been set to **true**.

Note: Auto-fitting the text size is not available on web clients.

▼ Bold Text

Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

▼ Italic Text

Select `true` or `false` from the dropdown list of the combo box. 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. 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
- 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 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.

▼ 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

▫ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the

current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Margin Bottom

  Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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** 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 control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

DateTime (iOS)

DateTime controls are available only on iOS client devices and are used to format dateTimes obtained from a node in a data 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

DateTime events

The **`OnFinishEditing event`** is available. For a description of the actions that can be defined for this event, see the Actions section.

DateTime properties

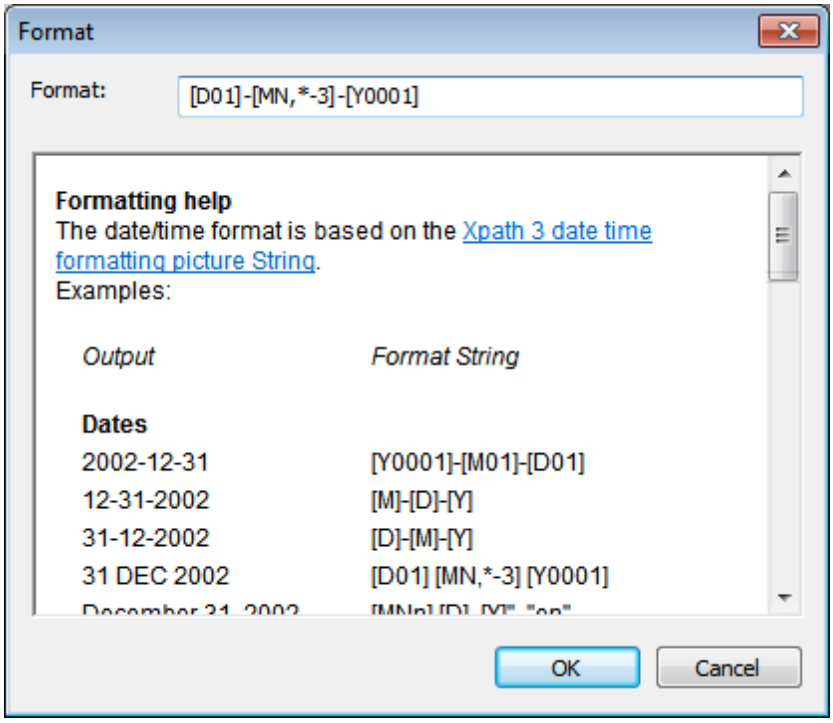
The control's properties are available in the Styles & Properties Pane, and are listed below in the order in which they appear.

Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ 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*).



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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** 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 control. (If there are multiple Assertion Message 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 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 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 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 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-in-words. 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`** 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'`**.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the

current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
  - 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 disabled, use the **Background Color (Disabled)** property.

- 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.

- Horizontal Alignment

  Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or

double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

  In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

  In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 an XML node, an association must be made in the design with this node (by dropping this node from the data 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.

Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Edit Field events

The **`OnTyping event`** is available. For a description of the actions that can be defined for edit field events, see the Actions section.

Edit Field properties

The control's properties are available in the Styles & Properties Pane, 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.

▼ 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 data 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. For some controls you can choose between the two methods of entering the property value: right-click the property and select the entry method you want from the context menu (fixed-value or XPath). For other controls, there is only one method of entering the property value.

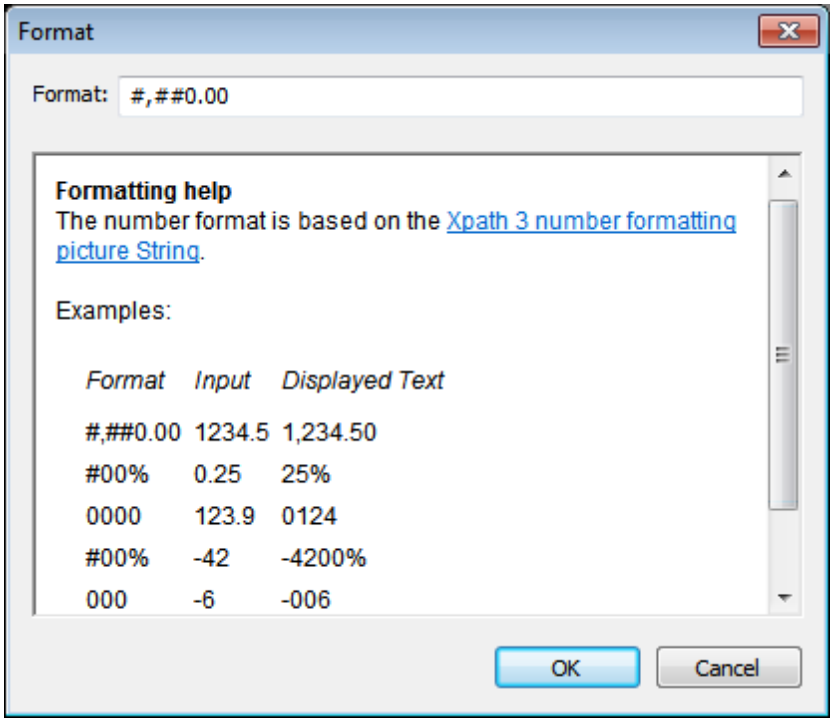
Note: The **`$MTControlValue`** variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

▼ Multiline

Sets multiline input/display on or off (`true/false`). The default is `false`.

▼ Number Format String

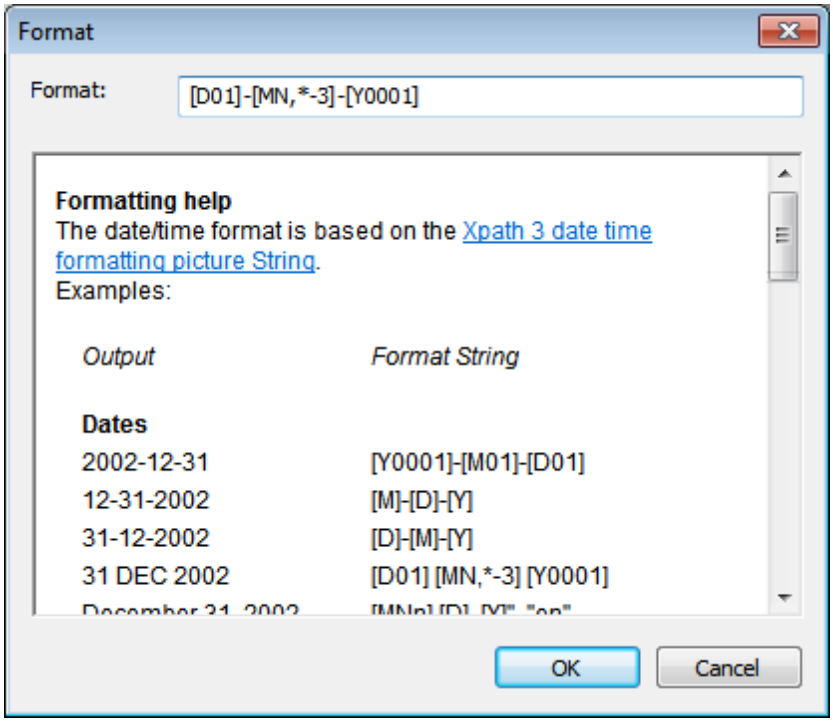
Click the **Additional Dialog** button and enter a number format in the Format dialog that appears (*screenshot below*).



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*).



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 During Typing

Control actions can be triggered as soon as typing starts. For example, the page can be scrolled to bottom when typing starts, or a node can be updated while typing. Property values are `true` or `false`. Default is `true`. For HTML browsers used as client devices, the value is always `false`.

Control Action

Click the **Additional Dialog** button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

- 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.

  Note: The **`$MTControlValue`** 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 control. (If there are multiple Assertion Message 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 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 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 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-in-words. 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`** 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'**.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

Bold Text

Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

Italic Text

Select `true` or `false` from the dropdown list of the combo box. 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. 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
- 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 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.

▼ 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 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Margin Left

  Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Margin Right

  Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

  In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($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** 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 control. 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*). See the section Style Sheets 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) to assign properties

specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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), select the property and click **Reset** in the pane's toolbar.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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, 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.

Line Width

Sets the width (thickness) in pixels of the line. Select a width value (in pixels) 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.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use

in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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.

Note: The **$MTControlValue** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Margin Bottom

  Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Style Sheet

  The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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, you can set the slider's minimum and maximum values (*see 'Horizontal Slider properties' below*), and specify control actions 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Horizontal Slider events

The **`OnSliding event`** is available. For a description of the actions that can be defined for Horizontal Slider events, see the Actions section.

Horizontal Slider properties

The control's properties are available in the Styles & Properties Pane, 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.

- 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.

- Trigger Control Actions During Sliding

  If set to `true`, control actions are triggered **during** sliding (either continuously or at specified intervals). When set to `true`, the property `Trigger Control Actions on Sliding Interval (ms)` appears; you can set the trigger intervals here.

  If set to `false`, control actions are triggered **after sliding has finished**. Default is `false`.

- 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 `1000 ms`. 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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

- 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.

Note: The **`$MTControlValue`** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the

resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Style Sheet

  The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 data 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. For example, if a combo box selection changes the selection of an image, a Reload action 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

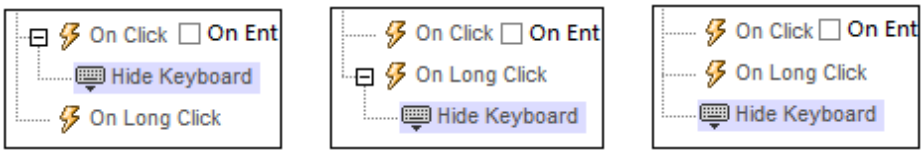
Image events

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.

OnImageClicked (OnClick, OnLongClick)

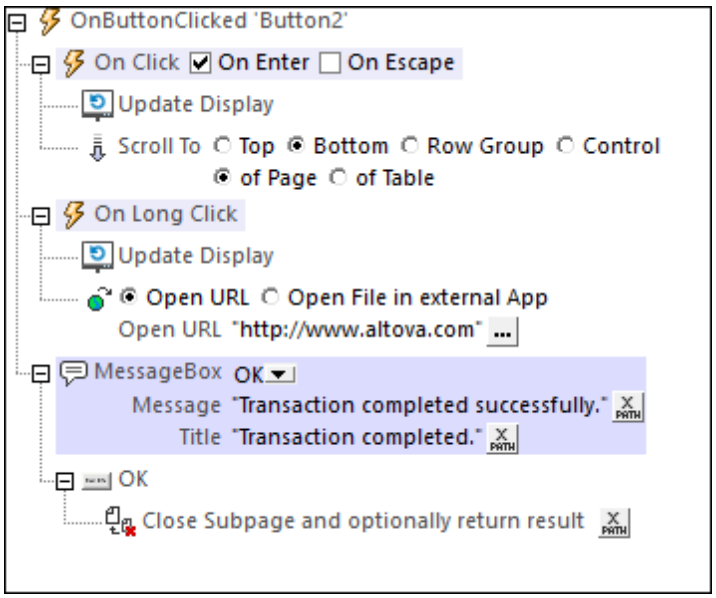
The end user can click the control in one of two ways: a short tap or a longer press. A sequence of different actions can be specified for each type of click. An additional sequence of actions can be performed after those of the end-user click. The additional sequence is

defined after the `On Long Click` event.



- **`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 for the different click events, for example, as shown in the screenshot below.



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` and/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 gets the focus and is "clicked" when the respective key (**Enter** or **Escape**) is tapped.

This setting can also be accessed via the control's `On Enter/Escape` property. For more details, see the description of the property below.

Note: These shortcut assignments can be viewed by selecting the **Page | Show/Define Tab Order** menu command.

Note: This feature is available only on Web clients and Windows clients.

Image properties

The control's properties are available in the Styles & Properties Pane, 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.

▼ 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.

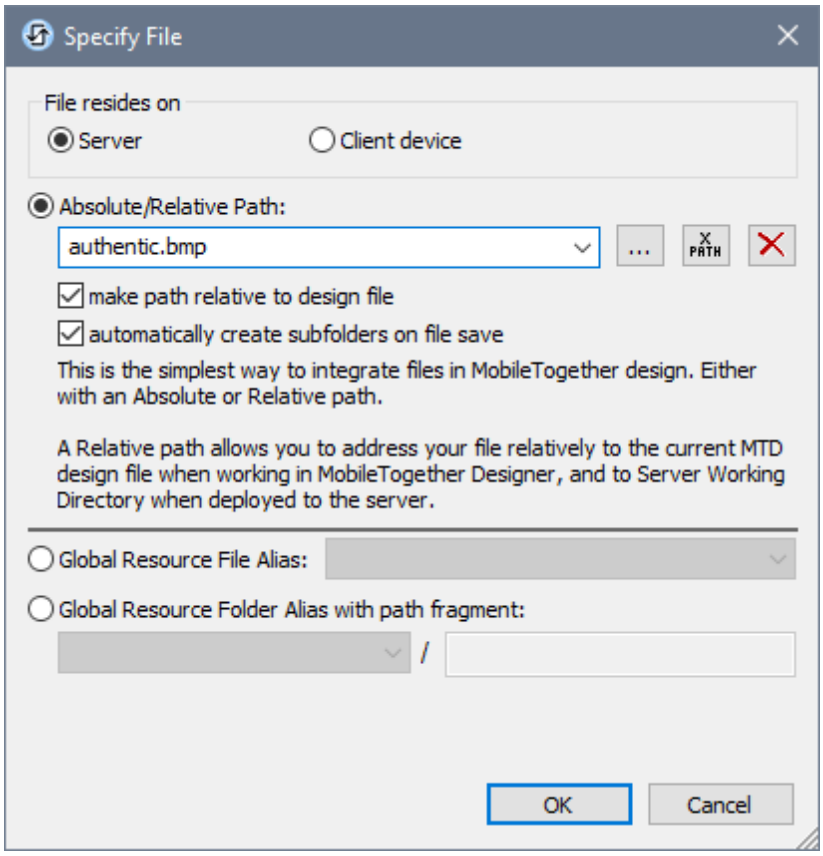
The type of image source is provided by the property `Image Source Type` (*see next property below*). By default, `Image Source Type` is set to `url`. The `Image Source` property automatically opens the corresponding dialog: Specify File dialog for `url` *(see below)*, and Edit XPath XQuery Expression dialog for `base64` (*see Base64-Encoded Images*).

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. For example, if a combo box selection changes the selection of an image, a Reload action 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.

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*).



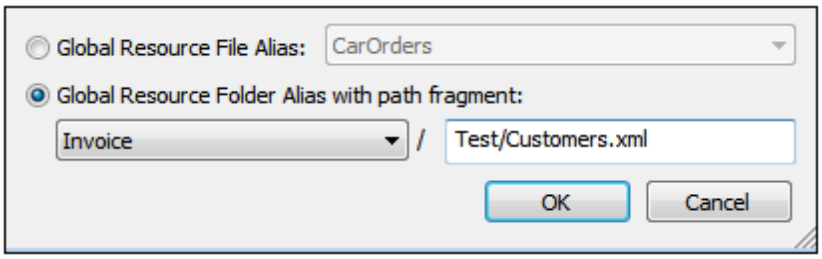
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design 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 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 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 for details. 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 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.

- *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**). See the section Altova Global Resources for details.

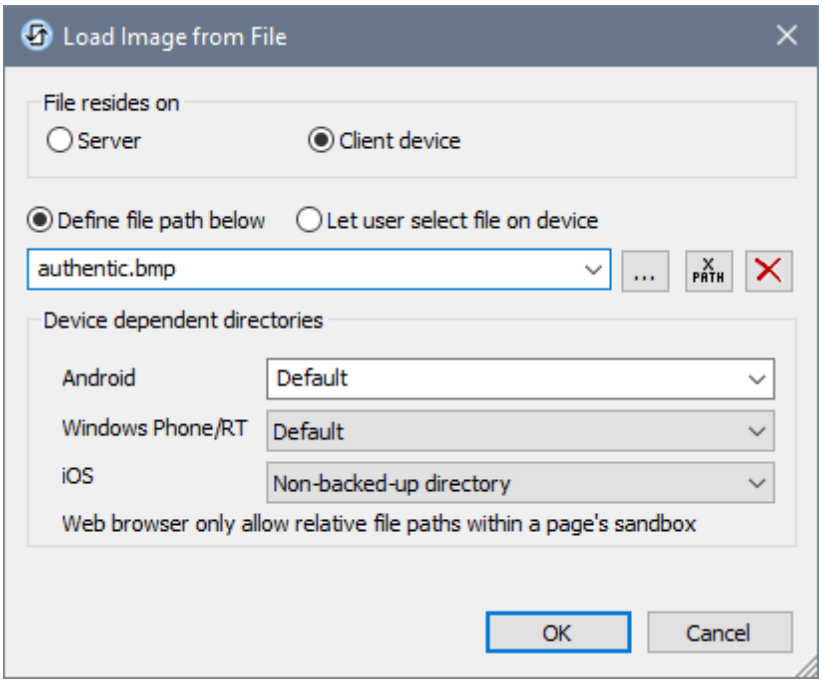
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



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**). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 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.



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 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 (*Source File* and *Target File* options), Load/Save File, Load/Save Image, and Load/Save Binary File.

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 (**Tools | Options**).

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 that is selected by the `Image Source` property above. Two type options are available:

- `url`: a binary image file, such as a `PNG` or `BMP` image file
- `base64`: a base64-encoded string

The default is `url`.

- Username

  Sets a user name for user access to the resource. Double-click in the property's value field to edit.

- Password

  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`.

- Embed Image

  This property is enabled if `Image Source Type` is `url`. 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). Once an image is embedded in this way, it can quickly be re-embedded by right-clicking the image in the design and selecting **Re-embed Image**. The source image file will be converted afresh and re-embedded in the design file. The need to re-embed an image would typically arise if the binary image file has been modified.

- Control Action

  Click the **Additional Dialog** button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

- 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.

Note: The **$MTControlValue** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

▫ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the

values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Margin Right

  Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Margin Top

  Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Margin Bottom

  Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- On Enter/Escape

  Takes one of three values:

  - `OnEnter`: Specifies that this control gets the focus and is "clicked" when the **Enter** key is tapped.
  - `OnEscape`: Specifies that this control gets the focus and is "clicked" when the **Escape** key is tapped.
  - None: No action when any key 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 Control Events).

Note: This feature is available only on Web clients and Windows clients.

- Style Sheet

  The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 one of the page data sources (this 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

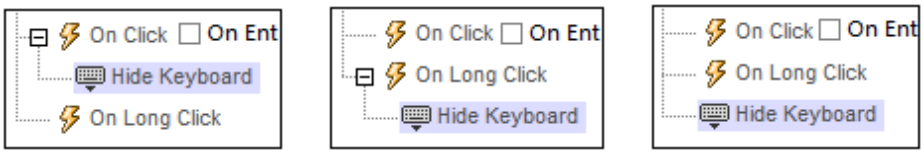
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **To 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.

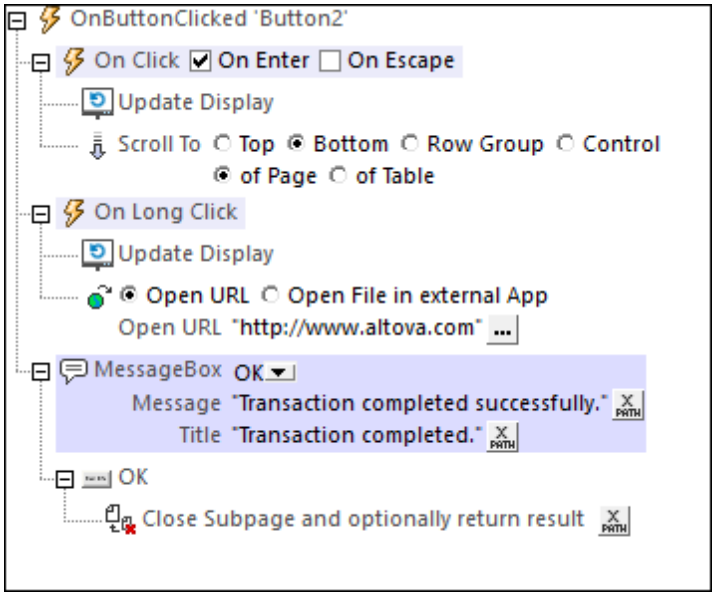
OnLabelClicked (OnClick, OnLongClick)

The end user can click the control in one of two ways: a short tap or a longer press. A sequence of different actions can be specified for each type of click. An additional sequence of actions can be performed after those of the end-user click. The additional sequence is defined after the `On Long Click` event.



- **`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 for the different click events, for example, as shown in the screenshot below.



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` and/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 gets the focus and is "clicked" when the respective key (**Enter** or **Escape**) is tapped.

This setting can also be accessed via the control's `On Enter/Escape` property. For more details, see the description of the property below.

Note: These shortcut assignments can be viewed by selecting the **Page | Show/Define Tab Order** menu command.

Note: This feature is available only on Web clients and Windows clients.

Label properties

The control's properties are available in the Styles & Properties Pane, 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.

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 data 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. For some controls you can choose between the two methods of entering the property value: right-click the property and select the entry method you want from the context menu (fixed-value or XPath). For other controls, there is only one method of entering the property value.

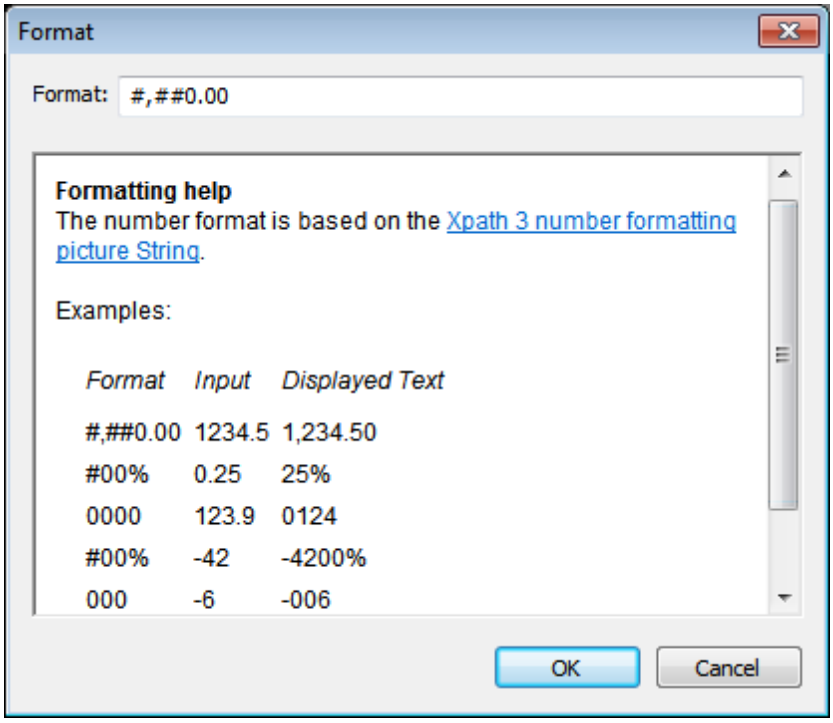
Note: The **`$MTControlValue`** variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

Multiline

Sets multiline input/display on or off (`true/false`). The default is `false`.

Number Format String

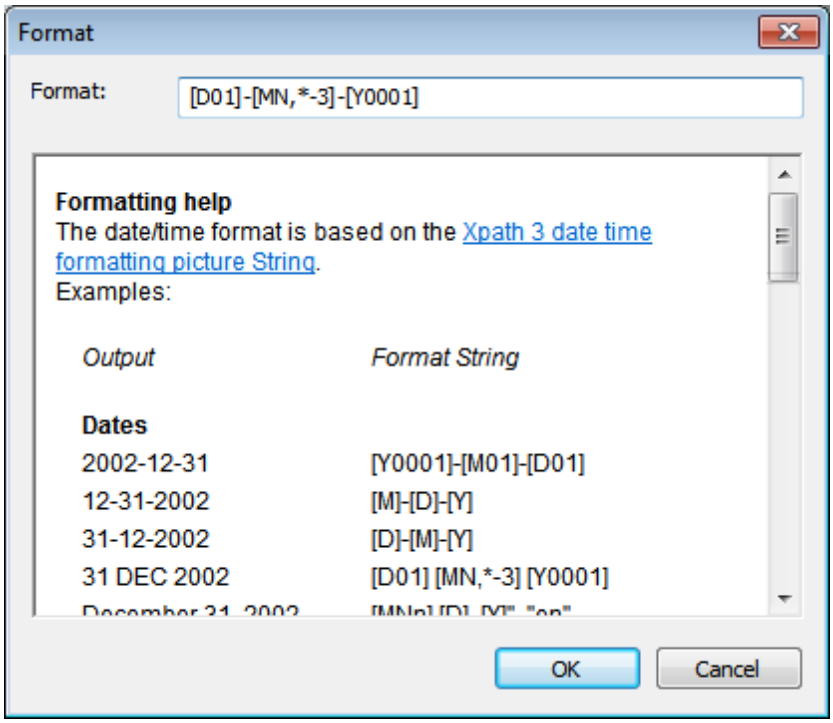
Click the **Additional Dialog** button and enter a number format in the Format dialog that appears (*screenshot below*).



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*).



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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** 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-in-words. 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`** 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'`**.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Text Size Auto-Fit

Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

The auto-fit function cannot be used if the **`Multiline`** property has been set to **`true`**.

Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
  - 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 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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

$MT_CanvasX * 0.5. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Maximum Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▫ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▫ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ On Enter/Escape

Takes one of three values:

- `OnEnter`: Specifies that this control gets the focus and is "clicked" when the **Enter**

key is tapped.
- `OnEscape`: Specifies that this control gets the focus and is "clicked" when the **Escape** key is tapped.
- None: No action when any key 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 Control Events).

Note: This feature is available only on Web clients and Windows clients.

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets for more information.

Browser CSS Class

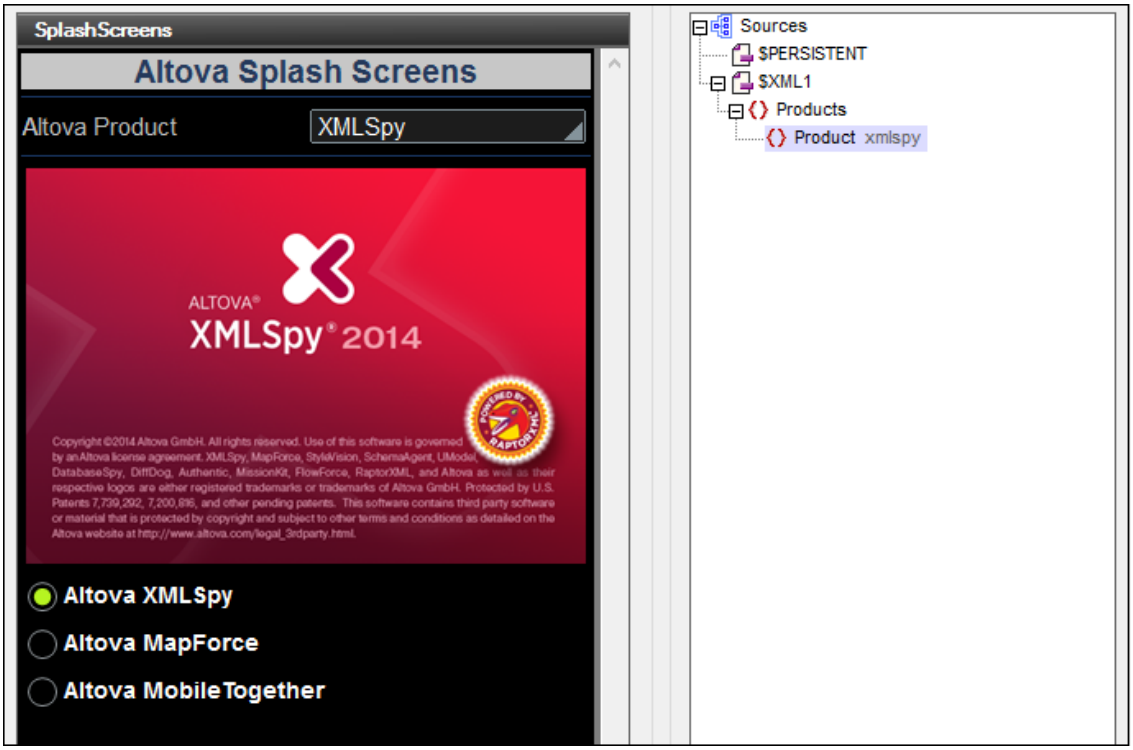
The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: **`LabelClassOne LabelClassTwo`**. These classes can then be used in a CSS file (specified in the Browser Settings dialog) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 data 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 data 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 XML data 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*).



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, 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.

- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Radio Button events

The **`OnFinishEditing event`** is available. For a description of the actions that can be defined for this event, see the Actions section.

Radio button properties

The control's properties are available in the Styles & Properties Pane, 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.

▾ 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 data 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. For some controls you can choose between the two methods of entering the property value: right-click the property and select the entry method you want from the context menu (fixed-value or XPath). For other controls, there is only one method of entering the property value.

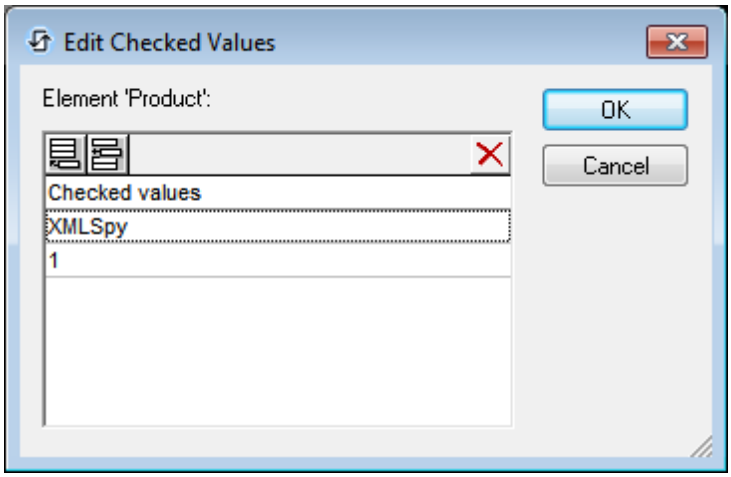
Note: The **`$MTControlValue`** variable is **not** available for the generation of the value of the `Text` property. If used, then a validation error results.

▾ Multiline

Sets multiline input/display on or off (`true/false`). The default is `false`.

Checked Values

Provides an XML data value for the selected state of the control. Click the **Additional Dialog** button to display the Edit Checked Values dialog (*screenshots below*). Enter a value for the checked (selected) state. If the radio button is selected by the end-user, the first value of the list will be entered as data in the XML node associated with the control.



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`** 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 at least one XML value. These XML values are defined in the `Checked Values` property.

The `Auto Correct Value` property has two possible values: `true` or `false`. If the property is set to `true`, XML values are automatically corrected to the values defined for the checked and unchecked states in the `Checked Values` property. For example, if the checked value has an XML value of `child` and the unchecked value has an XML value of `adult`, then, if the control is checked, the XML value will be corrected to `child` in case something else is entered. If the control is unchecked, the XML value will be `adult`. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction. A correction would be necessary, for example, if the control is associated with a node having content that is not a legitimate XML value for the current state of the control. 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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** 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-in-words. 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`** 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'`**.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Text Size Auto-Fit

  Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

  If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

  The auto-fit function cannot be used if the **`Multiline`** property has been set to **`true`**.

  Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
  - 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 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The 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).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Maximum Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The Maximum 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 value*: select a pixel 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. The default is wrap_content for all controls.

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▫ *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 **points** in the viewport coordinate

space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Margin Top

  Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Margin Bottom

  Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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** 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 control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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.

Notes

- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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` is available. For a description of the actions that can be defined for this event, see the Actions section.

Rich Text properties

The control's properties are available in the Styles & Properties Pane, 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.

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 data 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. For some controls you can choose between the two methods of entering the property value: right-click the property and select the entry method you want from the context menu (fixed-value or XPath). For other controls, there is only one method of entering the property value.

Note: The `$MTControlValue` 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.

Note: The **`$MTControlValue`** 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.

▼ 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 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.

▼ 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 dialog).

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Rich Text Height

Specifies the height of the Rich Text control. The default behavior is for the height of the control to increase as content increases, up to a maximum of the screen height. 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) for the control.

Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of

the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($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** 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 control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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 data source node. When the data 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	`$MT_PageBackgroundColor`	`Text Color`
Signature background-color	`mt-invert-color` (`$MT_PageBackgroundColor`)	`Background Color`
Signature image width	`min ($MT_CanvasX, $MT_CanvasY)`	`Signature Creation Width`
Signature image height	`min ($MT_CanvasX, $MT_CanvasY) div 2`	`Signature Creation Height`

The main settings for the signature field are:

- a page source link, which is the data source node where the signature image data is stored. Drag a data 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 data source; till such an action is executed, the data is stored only in the temporary XML tree

Notes

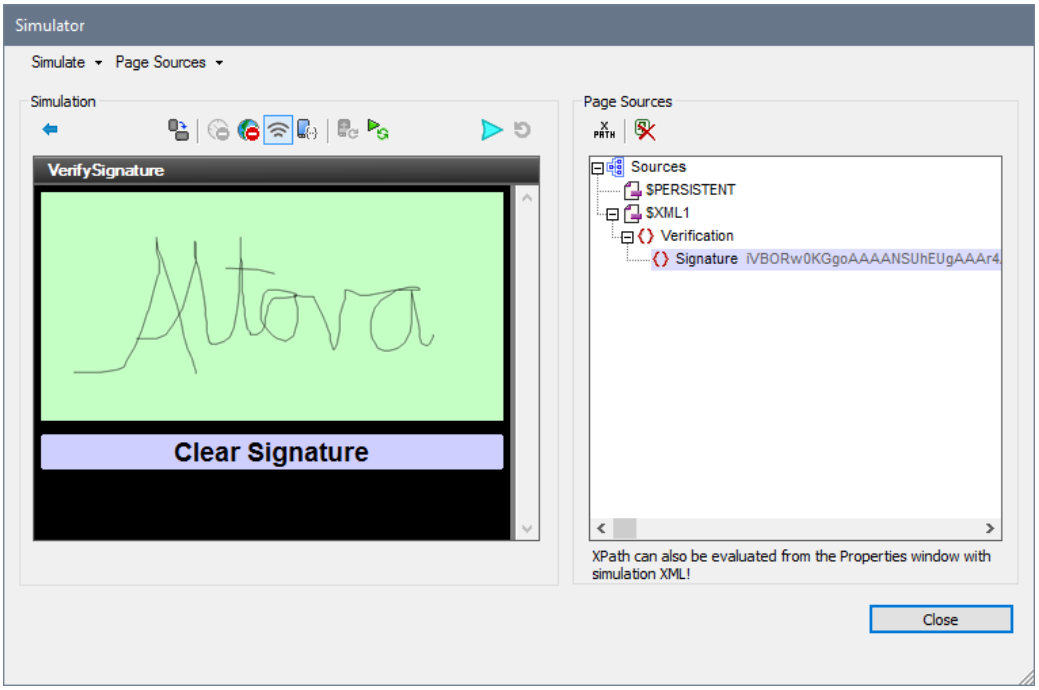
- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the data source tree are displayed in a bold font. Tree nodes that are not page source links are displayed 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.

- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Enabling the end user to edit a signature

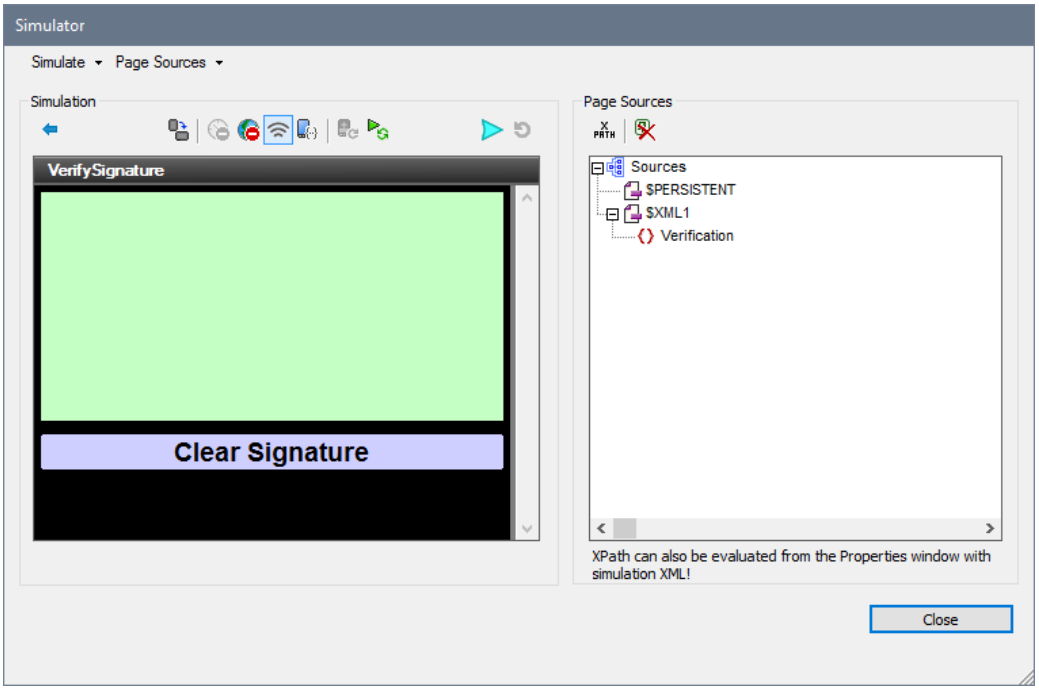
The end user's signature is created as an image in a page source node. This being the case, an end user can only add data to a signature drawing that has been started. If the image needs to be edited—for example, if the end user has drawn the signature unsatisfactorily—then the image data must be removed 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 action as the button's `OnClick` event, and set the signature's page source node as the node to delete.



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.



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 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 Filed control.

Signature Field properties

The control's properties are available in the Styles & Properties Pane, 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.

▼ 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.

Note: The **`$MTControlValue`** 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
- 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 disabled, use the **`Background Color (Disabled)`** property.

▼ 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, 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).

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Signature Creation Height

A number, in pixels, 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).

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

  In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For

these, the default is `wrap_content`.

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

  In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel 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. The default is `wrap_content` for all controls.

Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

- Margin Left

  Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

    In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

- Margin Right

  Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or

double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

  In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets for more information.

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), select the property and click **Reset** in the pane's toolbar.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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, 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.

Space Height

Sets the height in pixels of the selected space object. Select a height value (in pixels) 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.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate

space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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.

  Note: The **$MTControlValue** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

- Style Sheet

  The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

Switch

A switch is associated with a data 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Switch events

The **`OnFinishEditing event`** is available. For a description of the actions that can be defined for this event, see the Actions section.

Switch properties

The control's properties are available in the Styles & Properties Pane, 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

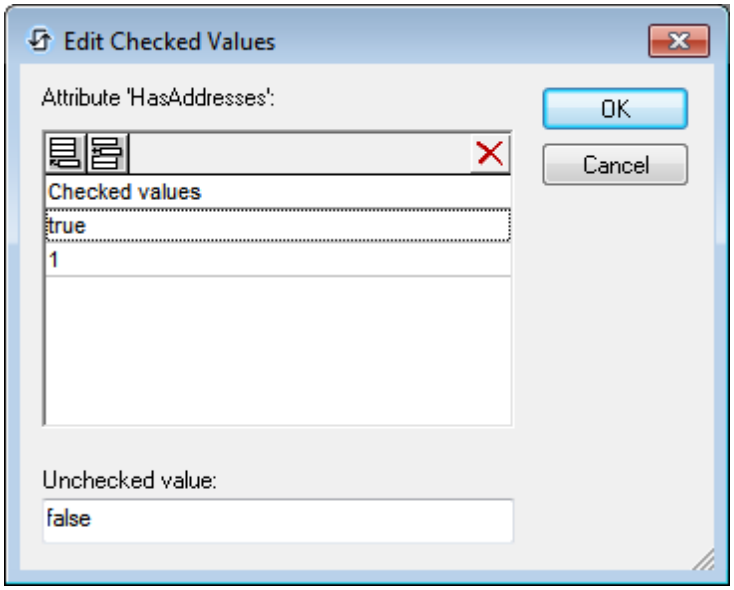
inside the value field to edit.

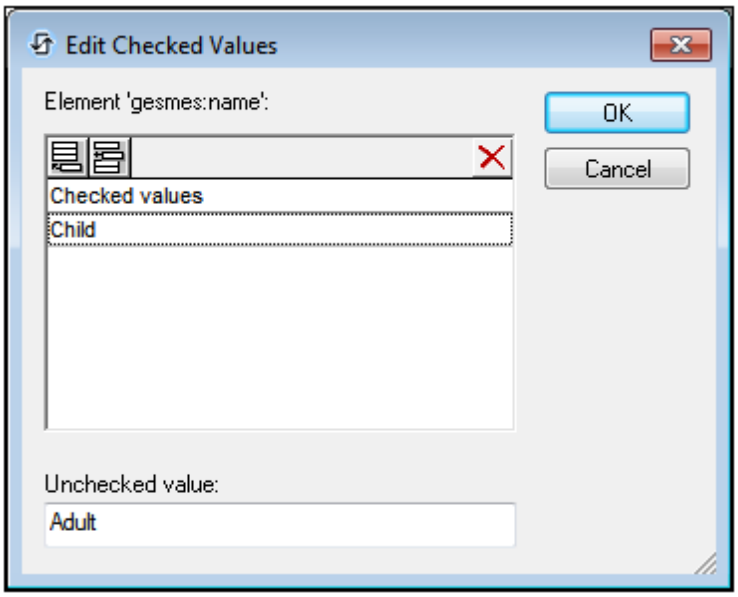
- Multiline

  Sets multiline input/display on or off (`true/false`). The default is `false`.

- Checked Values (true/false)

  Provides an XML data value for the selected/unselected state of the control. Click the **Additional Dialog** button to display the Edit Checked Values dialog (*screenshots below*). Enter a value for the checked (selected) and unchecked (unselected) state. The value corresponding to the control state (selected/unselected) chosen by the end-user will be entered as data in the XML node associated with the control. To specify which node will receive the value, make a page source link from the check box to an XML data source node (by dragging the node on to the check box).





- 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`** 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 at least one XML value. These XML values are defined in the `Checked Values` property.

  The `Auto Correct Value` property has two possible values: `true` or `false`. If the property is set to `true`, XML values are automatically corrected to the values defined for the checked and unchecked states in the `Checked Values` property. For example, if the checked value has an XML value of `child` and the unchecked value has an XML value of `adult`, then, if the control is checked, the XML value will be corrected to `child` in case something else is entered. If the control is unchecked, the XML value will be `adult`. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction. A correction would be necessary, for example, if the control is associated with a node having content that is not a legitimate XML value for the current state of the control. 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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

▼ 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.

Note: The **`$MTControlValue`** 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-in-words. 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`** 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'`**.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate

space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Text Size Auto-Fit

  Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

  If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

  The auto-fit function cannot be used if the **Multiline** property has been set to **true**.

  Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
  - 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 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Maximum Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 the image width would be

`concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the `$MT_CanvasX` and `$MT_CanvasY` variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values

returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the

values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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** 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 control. 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*). See the section Style Sheets for more information.

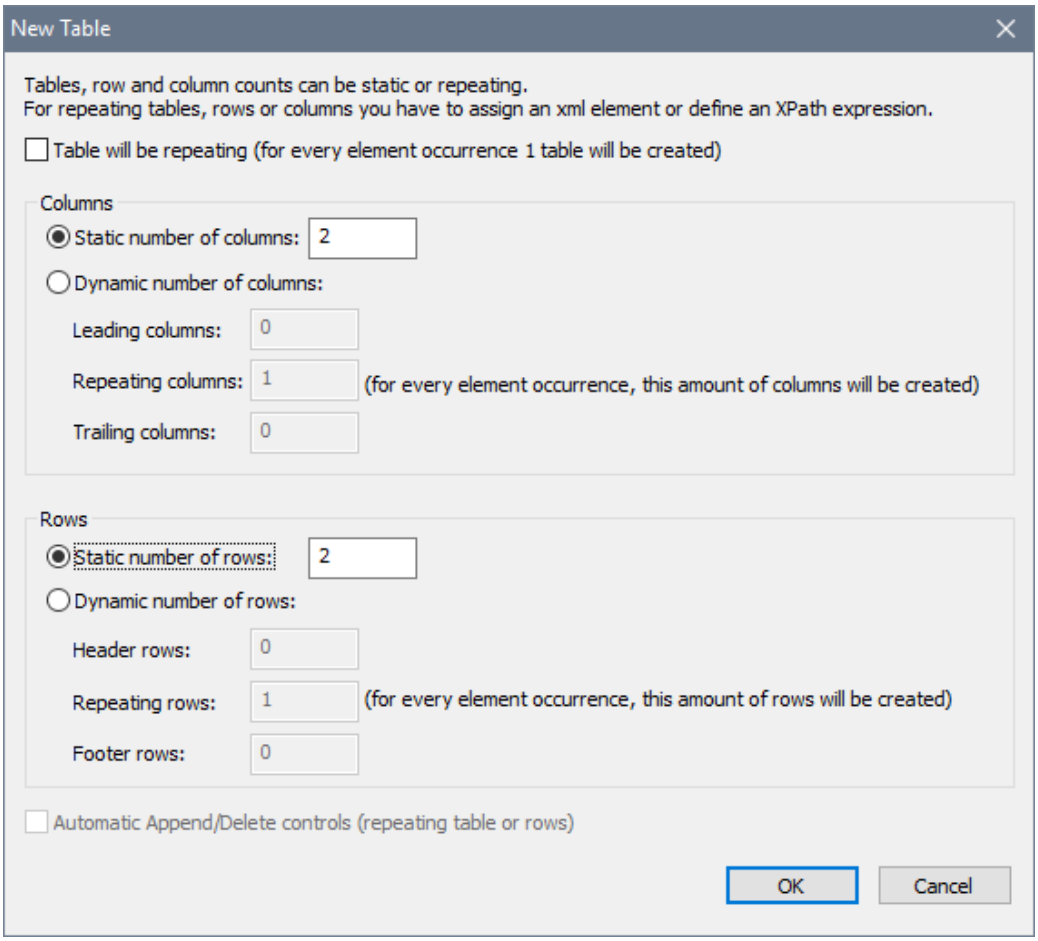
Browser CSS Class

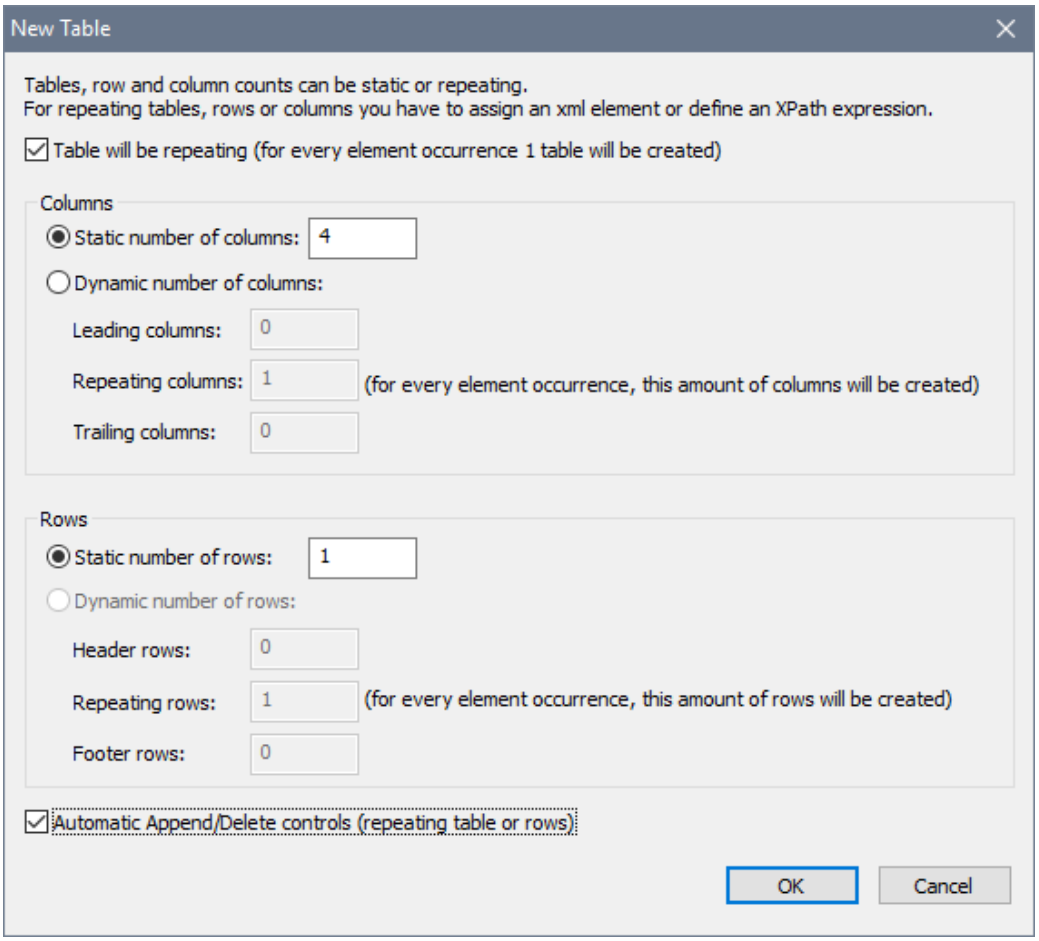
The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: **LabelClassOne LabelClassTwo**. These classes can then be used in a CSS file (specified in the Browser Settings dialog) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

Table

The Table control inserts a static table, repeating table, a table with dynamic rows, or a table with dynamic columns. 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 is created.
- A repeating table 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, 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, 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, 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 generate for repeating tables as well as for tables with dynamic 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 data source
- A page control
- A nested table

Table formatting properties are available in the Styles & Properties Pane and in the context menu of the table.

For more information, see the section Tables.

Notes

- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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, and are listed below in the order in which they appear.

Table Cell

▼ 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
- 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 disabled, use the **`Background Color (Disabled)`** property.

▼ Spans Column Groups

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 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** command 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 associated with the **week** element (which, in the data 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 data 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` 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. For details of how to span columns, see the section Row/Column joining and spanning.

▼ 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

Table Column

▼ 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 `Maximum Width` becomes available
- *percent value*: a percentage of the parent object; select a value from the dropdown list, or enter a value directly
- *pixel value*: select a pixel value from the dropdown list, or enter a value directly

▫ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Maximum Width

This property is available only when the table column's **Width** property has been set to **wrap_content**. The `Maximum 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 value*: select a pixel 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.

Note: The **$MTControlValue** 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

- 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 disabled, use the **`Background Color (Disabled)`** property.

▼ 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

▣ *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

▼ 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.

Note: The **`$MTControlValue`** 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
- 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 disabled, use the **`Background Color (Disabled)`** property.

- 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

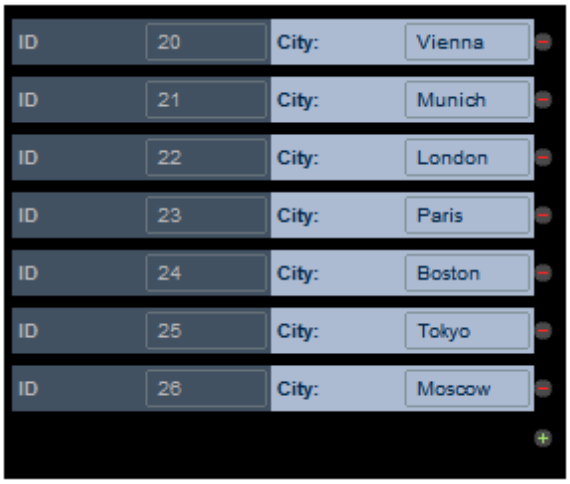
- *Note about background colors*

  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 display structures. The **Append** button in a solution appends another occurrence of the repeating structure. The **Delete** button in a solution 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.



  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 value can be changed at any time.

- Create For Each Item In

  Sets the number of times the table 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 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 table repeats is automatically modified. For example, if the property's XPath expression is `$XML1/Office/Department`, then the number of times that the table is generated will be equal to the number of `$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.

Note: The **`$MTControlValue`** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

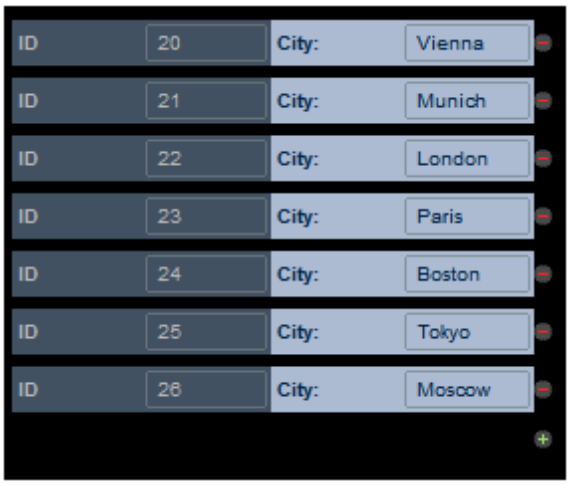
Table

▼ Repeating

Defines whether the table is repeating or static. 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.

▼ Automatic Append/Delete Controls

Automatically adds append/delete controls to display structures. The **Append** button in a solution appends another occurrence of the repeating structure. The **Delete** button in a solution 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.



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 value can be changed at any time.

▼ Create For Each Item In

Sets the number of times the table 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 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 table repeats is automatically modified. For example, if the property's XPath expression is `$XML1/Office/Department`, then the number of times that the table is generated will be equal to the number of `$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()`

▼ Maximal Table Width

The **`Maximal Table Width`** 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.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

For more information, see Table Properties.

▼ Maximal Table Height

The **Maximal Table Height** property specifies the table's height on the device's screen, 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. 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 **Maximal 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 can be displayed.
- *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.

⊟ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

For more information, see Table Properties.

▼ Scroll Vertically

The **Scroll Vertically** property becomes available after a value has been set for the **Maximal 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 **Maximal Table Height** 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.

▼ 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 **Maximal Table Height** 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.

For more information, see Table Properties.

▼ 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.

Note: The **$MTControlValue** 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
- 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 disabled, use the **`Background Color (Disabled)`** property.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The 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).

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 the image width would be

`concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the `$MT_CanvasX` and `$MT_CanvasY` variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values

returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the

values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

Table Padding

The `Table Padding` property sets whether padding is used for tables on iOS devices. The default value is **true**. If table padding is used, then the table is given a padding of **9px** on right and left, and **5px** on top and bottom.

Style Sheet

The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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.

Time

Time controls are used to format times obtained from a node in a data 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane have priority.

Time (control) events

The **`OnFinishEditing event`** is available. For a description of the actions that can be defined for this event, see the Actions section.

Time properties

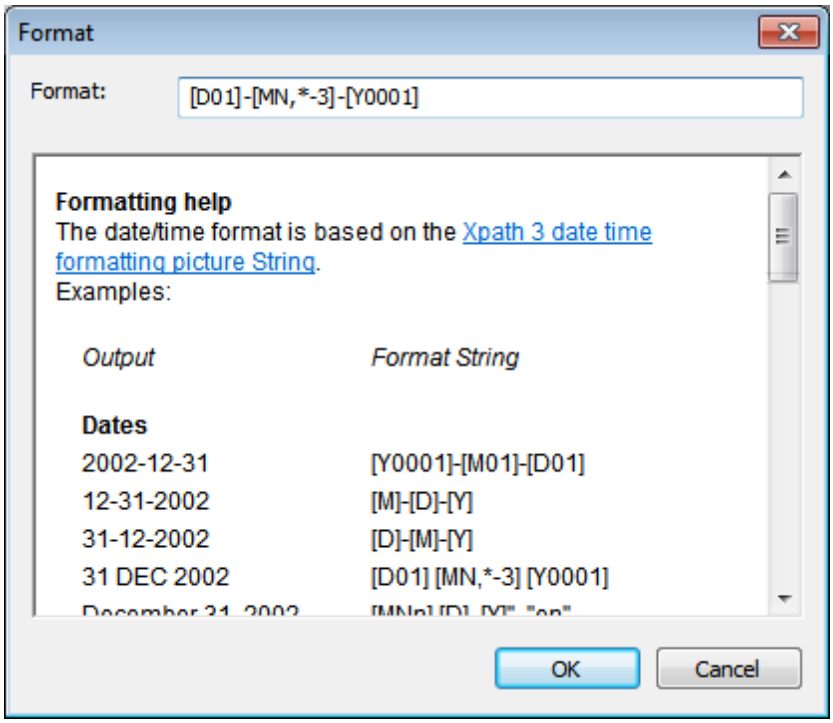
The control's properties are available in the Styles & Properties Pane, and are listed below in the order in which they appear.

Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ 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*).



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. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

▼ 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.

Note: The **`$MTControlValue`** 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 control. (If there are multiple Assertion Message 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 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 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 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-in-words. 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`** 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'`**.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

- Text Size Auto-Fit

  Sets whether or not the text size should be reduced to fit the width of the control. The property's values are: `Individually (true)` | `Off (false)` | `Group X`. The default is `Off (false)`. In Design View, text size can be reduced to a minimum size that is 50% of the font size.

  If set to one of the nine groups, then that control belongs to a group of controls that is identified by its group number. To add a control to a group, select the control and assign it to the group. The text size of all the controls in a group will be auto-resized to a size that is the same as that of the smallest font size in the group after calculating sizes for auto-fit. This property can thus ensure 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.

  The auto-fit function cannot be used if the **Multiline** property has been set to **true**.

  Note: Auto-fitting the text size is not available on web clients.

- Bold Text

  Select `true` or `false` from the dropdown list of the combo box. You can also use an XPath expression. Default is `false`.

- Italic Text

  Select `true` or `false` from the dropdown list of the combo box. 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. 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
- 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 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.

▼ 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.

▼ Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

▼ 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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

Maximum Control Width

This property is available only when the control's **`Control Width`** property has been set to **`wrap_content`**. The `Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or

double-click in the value field to enter a pixel length.

- *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

  In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($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** 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 control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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), select the property and click **Reset** in the pane's toolbar.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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, 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.

Line Width

Sets the width (thickness) in pixels of the line. Select a width value (in pixels) 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.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the

values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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.

  Note: The **$MTControlValue** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

- Horizontal Alignment

  Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

- Margin Left

  Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

  - *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Style Sheet

The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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.

Note the following features:

- Video resources are located via URLs. The URLs can be specified via a page source link, 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.
- 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 (for example, on a Button).
- 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 can be use to create custom buttons.
- The Video action provides functionality to start, pause, resume, stop, and skip to a specific time in the playback, as well as to play a time-defined segments.
- You can also specify a set of actions to perform when a video event is triggered (*see below*).
- A number of MobileTogether XPath extension functions 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`** 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) displays the associated node in a popup.
- All page source links in the data 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) and click **Delete Page Source Link**.
- To reset a style or property (in the Styles & Properties Pane), select the property and click **Reset** in the pane's toolbar.
- 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.
- To edit the XPath expression of a style or property (in the Styles & Properties Pane), select the style or property, and click **Edit XPath** in the pane's toolbar.
- **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).
- A control's CSS properties can be defined in the Styles & Properties Pane and/or in an external CSS file. Those defined in the Styles & Properties Pane 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) 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`**. 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*) or paused.

Video properties

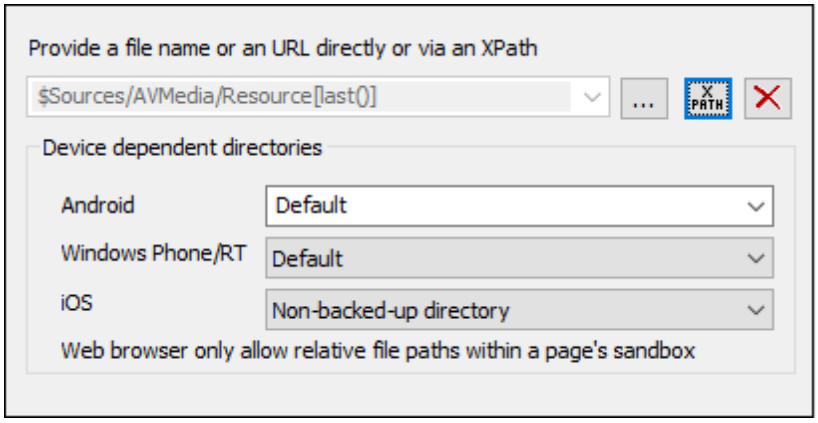
The control's properties are available in the Styles & Properties Pane, and are listed below in the order in which they appear.

▼ Name

The name is used to reference the control from elsewhere in the page or project. Double-click inside the value field to edit.

▼ 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.



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.

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. 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 action to load the binary audio/video data to a data source node; (ii) use the Save Binary action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 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

Sets a user name for user access to the resource. Double-click in the property's value field to edit.

▼ Password

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. 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.

▼ Control Action

Click the **Additional Dialog** button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's events are predefined and are each shown in a tab in the right-hand pane of the Actions dialog. 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.

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.

Note: The **`$MTControlValue`** variable is **not** available for the evaluation of the `Visible` property. If it is used, then a validation error results.

Horizontal Alignment

Sets the horizontal alignment to `left`, `center`, or `right`. Default is `center`. The value can also be specified via an XPath expression (which enables the value to be generated dynamically).

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 `Maximum 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 value*: select a pixel 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. The default is `fill_parent` for all controls except the `Image` and `Chart` controls. For these, the default is `wrap_content`.

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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Maximum Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The Maximum 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 value*: select a pixel value from the dropdown list, or enter a value directly

▼ Initial Width

A value in pixels 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**.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values

returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Initial Height

A value in pixels 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 * 9) div 16**.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin

Sets the control's margin offsets relative to the surrounding objects or to the borders of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic

variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Left

Sets the control's left margin offset relative to the object to the left or to the left border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($MT_CanvasX * 0.5, 'px')**.

▼ Margin Right

Sets the control's right margin offset relative to the object to the right or to the right border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the

resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Top

Sets the control's top margin offset relative to the object above it or to the top border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 `$MT_CanvasX * 0.5`. The XPath expression for the image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Margin Bottom

Sets the control's bottom margin offset relative to the object below it or to the bottom border of the containing object. Select a value in pixels from the dropdown list of the combo box, or double-click in the value field to enter a pixel length.

▣ *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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **`$MT_CanvasX`** and **`$MT_CanvasY`** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **`$MT_CanvasX`** and **`$MT_CanvasY`** variables. 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 **`$MT_CanvasX * 0.5`**. The XPath expression for the image width would be **`concat($MT_CanvasX * 0.5, 'px')`**.

- Style Sheet

  The `Style Sheet` property sets the style sheet to use for the control. 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*). See the section Style Sheets 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) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

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)** 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	– none –
Button	OnButtonClicked
Chart	OnChartClicked
Check box	OnFinishEditing
Combo box	OnFinishEditing
Date	OnFinishEditing
DateTime (iOS)	OnFinishEditing
Edit field	OnTyping
Horizontal Line	– none –
Horizontal Slider	OnSliding
Image	OnImageClicked
Label	OnLabelClicked
Radio button	OnFinishEditing
Rich Text	OnFinishEditing
Signature field	– none –
Space	– none –
Switch	OnFinishEditing
Table	– none –
Time	OnFinishEditing
Vertical Line	– none –
Video	OnVideoStarted, OnVideoError, OnVideoCompleted

Defining the actions of a control event

Action/s of a control event are defined in the Actions dialog (*see screenshot below*). Drag the desired action from the left-hand pane into the event tab in the right-hand pane.

The Actions dialog 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, click the **Additional Dialog** button of the `Control Action` property
- Click **Page | Actions Overview** to display the Actions Overview dialog. Then click the **Additional Dialog** button of the control event you want to define.

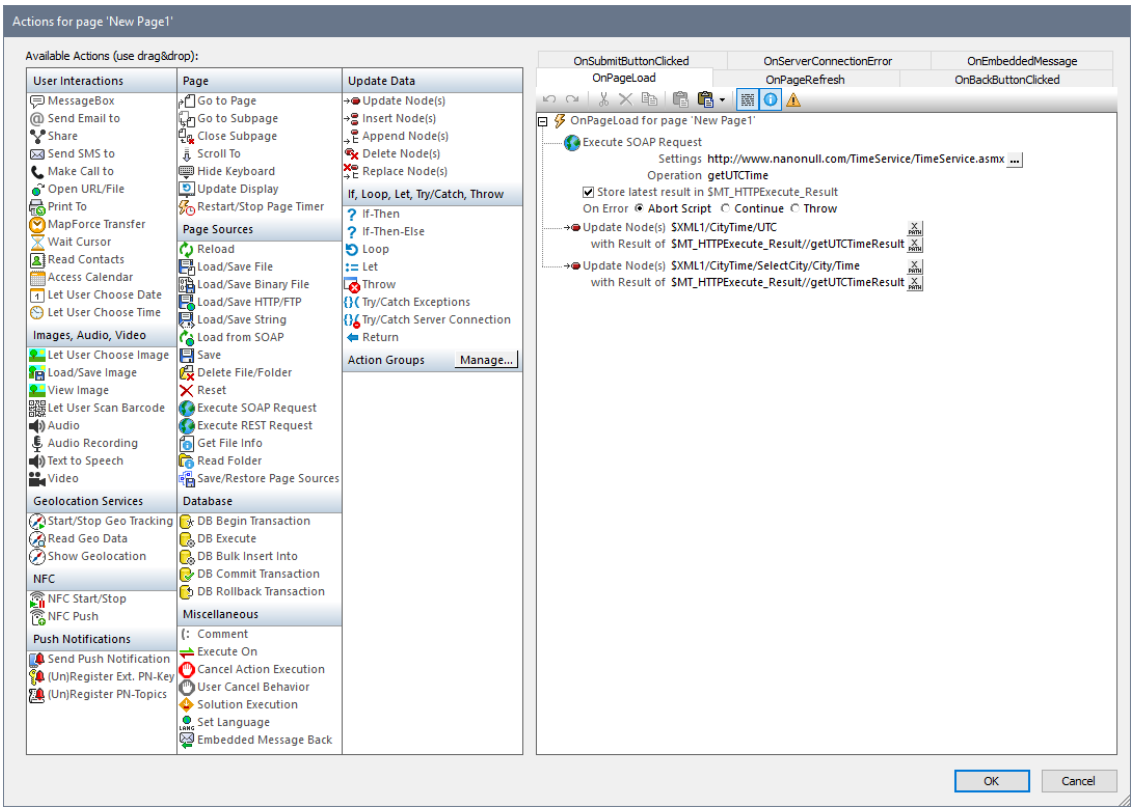
For more information about the various types of available control actions, see the section Actions.

Chapter 10

Actions

10 Actions

Actions can be defined for control events and page events. 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. The right-hand pane contains the page or control events. In the right-hand pane, each event is shown in a separate tab, and each event's actions are defined in the event's tab.



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, and the `Cancel` option triggers the End Solution action. 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 selected in the event pane	Result
Ctrl+double-click an 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 an 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

Showing all 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 **Show Usages of this Action in Messages Log**. All usages of that action in the design will be displayed in the Messages pane. The usages are listed by page, and then by control.

Action handling

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 so as to minimize switching between server and client for processing.

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.

Suppressing logs recursively

Logs are the messages that are displayed in the Messages Pane. 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 action.

	*No overlay*
	*Logs suppressed*
	*Logs force-enabled*

Event pane toolbar

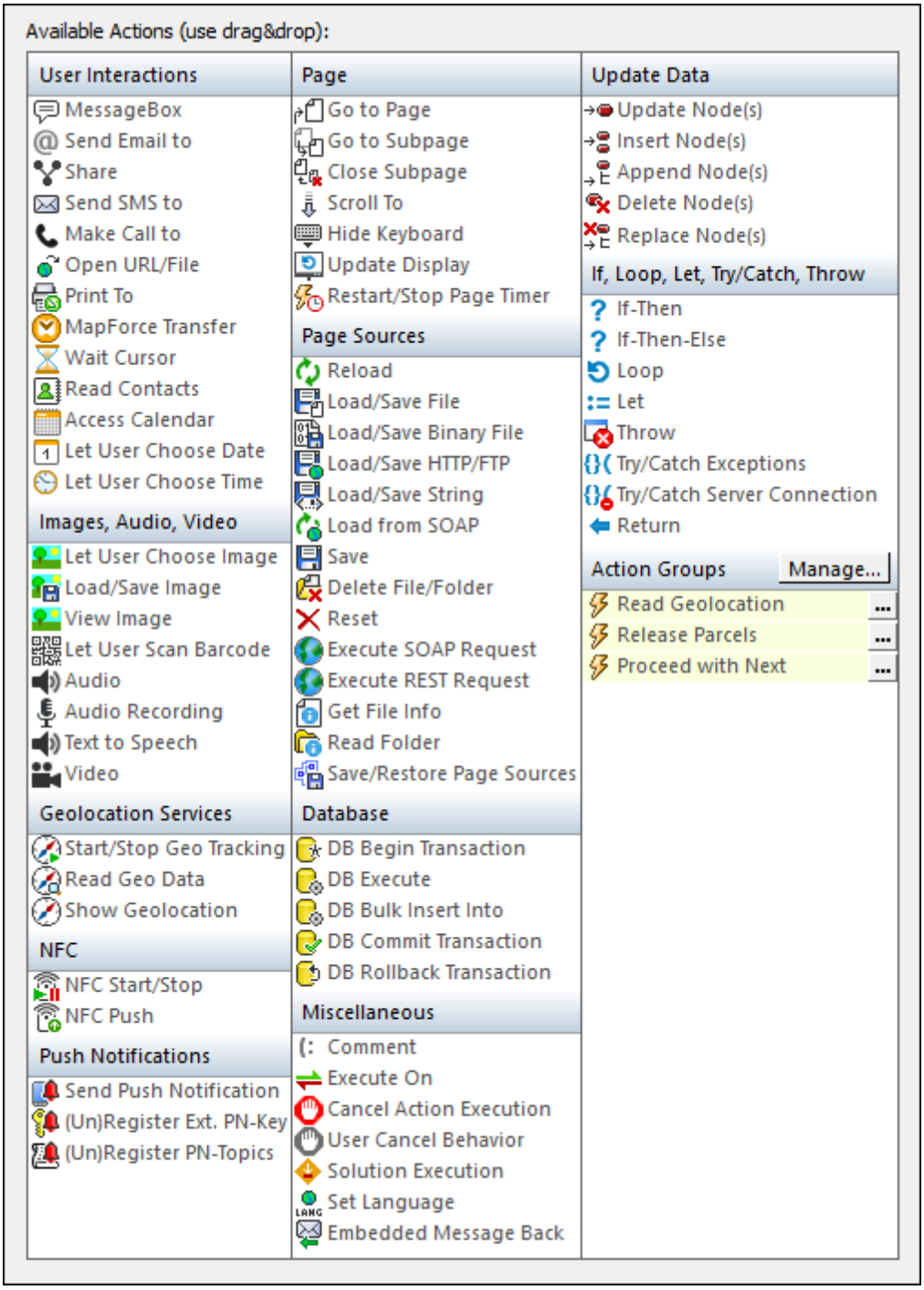
The Event Pane toolbar (*screenshot below*) offers the following commands, starting from the left:

- *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 Statements Multiline:* Toggles on/off multiline display of statements, such as XPath expressions
- *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
- *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

10.1 User Interactions

The following actions are available in the User Interactions group of the Actions dialog (*screenshot below*):

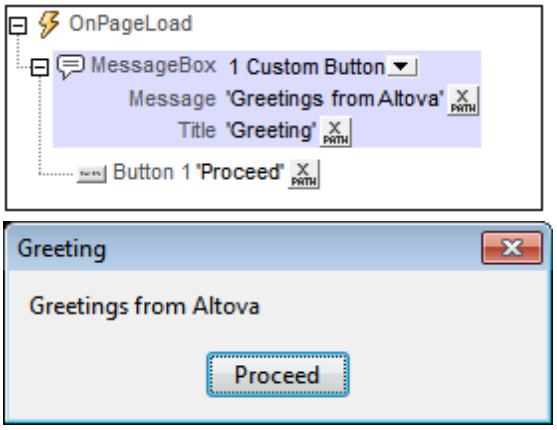
- Message Box
- Send Email To
- Share
- Send SMS To
- Make Call To
- Open URL/File
- Print To
- MapForce Transfer
- Wait Cursor
- Read Contacts
- Access Calendar
- Let User Choose Date
- Let User Choose Time



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 and Control Events*.

Message Box

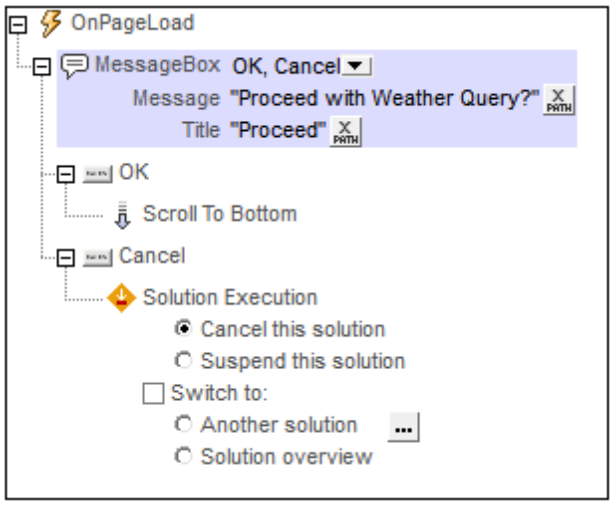
Defines a message box that is displayed when the event is triggered. The combo box enables you to select the buttons that appear in the message box. Predefined buttons are `OK`, `Yes`, `No`, `Cancel`. You can also define from one to three custom buttons. The text of custom buttons is specified in an XPath expression (for example, in the screenshot below left it is: **"Proceed"**).



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.

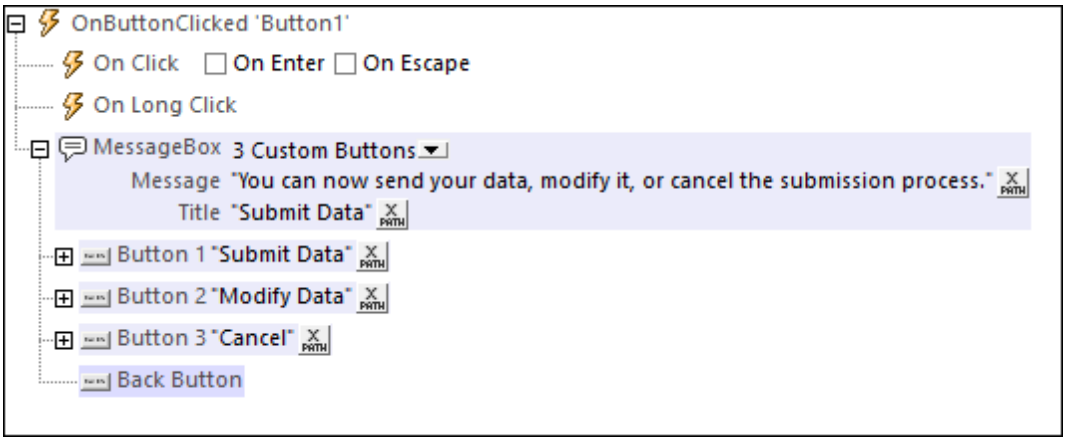
Message boxes with multiple user options

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.



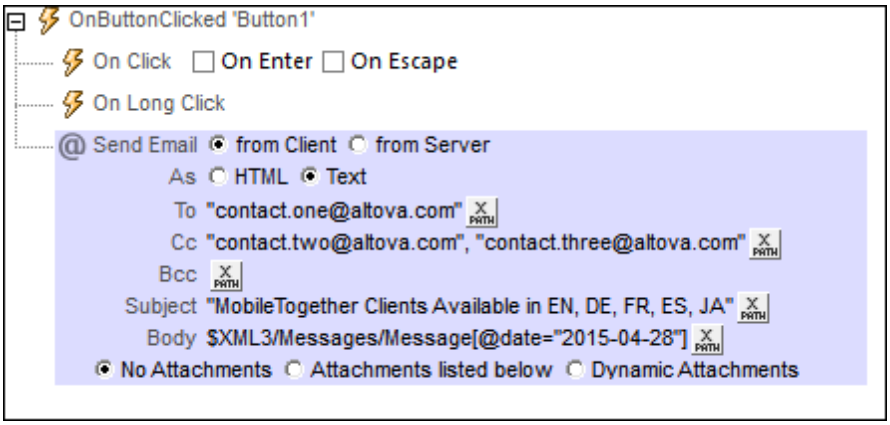
Custom buttons

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.



Send Email To

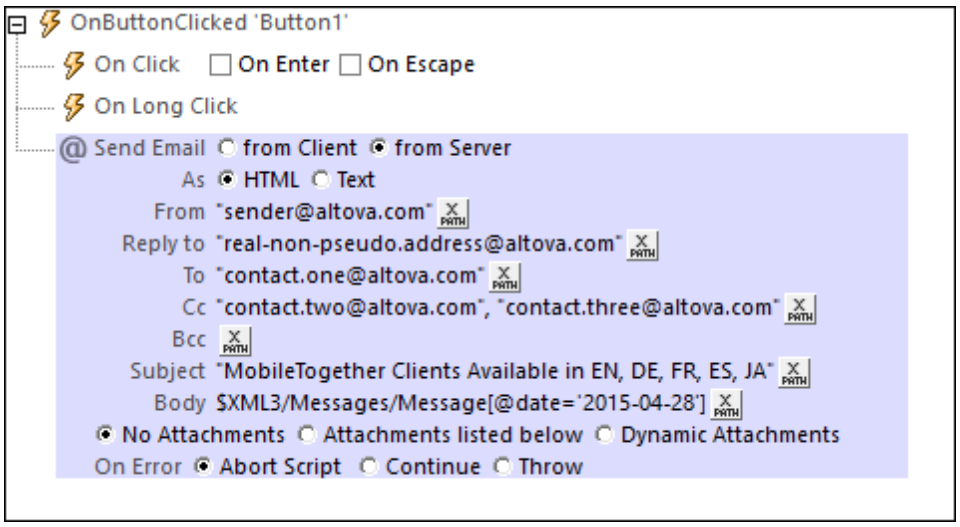
Sends an email from the email application on the mobile device or silently via the server to one or more recipients. The email can be sent as HTML or text. You can specify the recipients, the subject, and the message of the email body via XPath expressions. Additionally, text and image attachments can be generated. The settings of the Send Email To action are shown in the screenshot below and are described further below.



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 or updating the server settings on a client device via the link.

▼ 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 *As* field, the *From* field, and the *OnError* action.



- *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. Whether the email is sent as HTML or text depends on the client, and cannot be specified in the design.
- *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 or continued.

▼ 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 data 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 data 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")`.

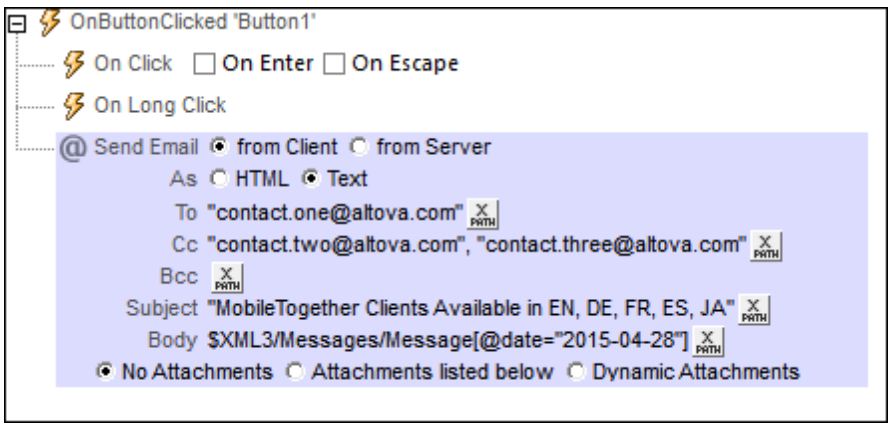
▼ From and Reply To

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 data 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*).



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.

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** 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 and Share 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.

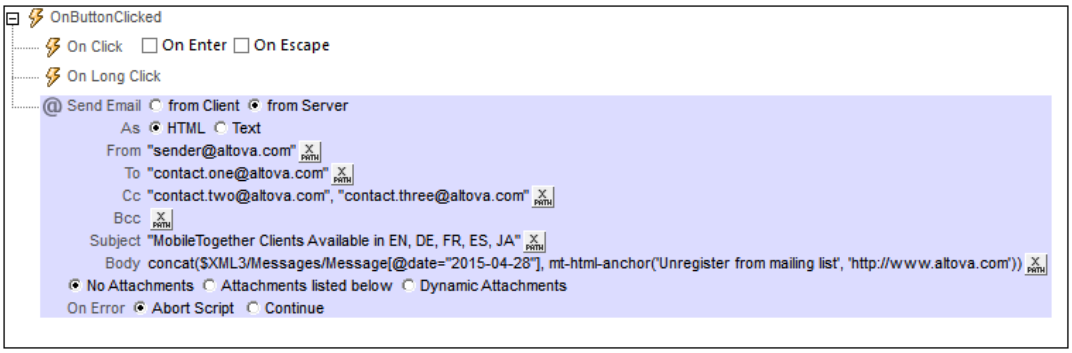
⊟ *Examples*

- `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`

Note: Attachments work with Android and iOS clients only.

▼ 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** function in the XPath expression of the *Body* option (*see screenshot below*).



The **mt-html-anchor** function takes two arguments: **LinkText** and **targetURL**. It uses these two arguments to create an HTML hyperlink element: **LinkText**

For example:

```
mt-html-anchor('Unregister from mailing list', 'http://www.altova.com')
```

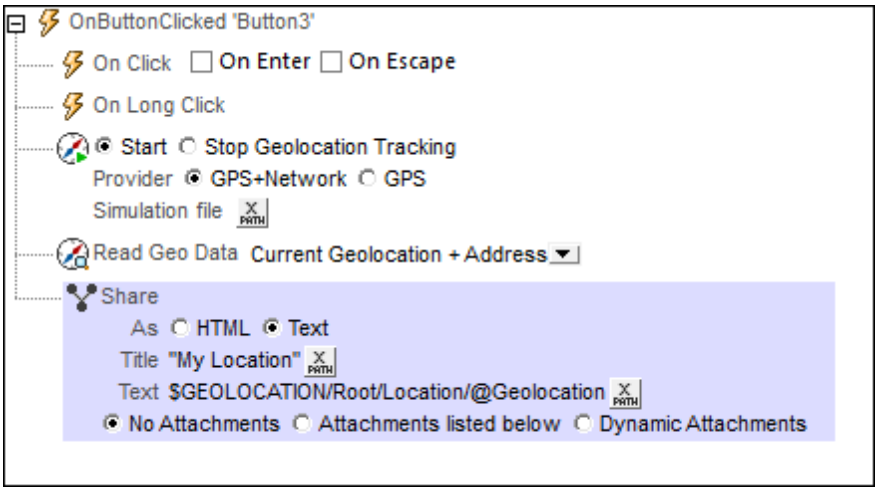
generates the HTML code fragment:

```
<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.

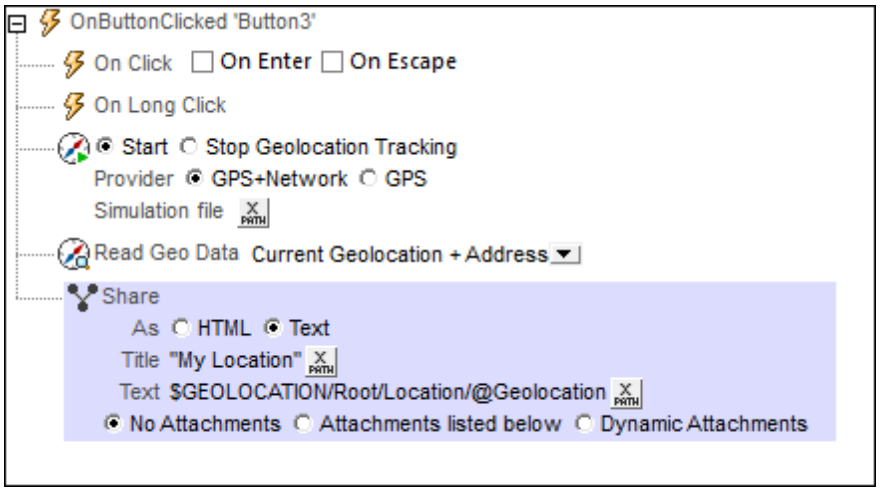
Share

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.



▼ 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 data sources.



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 and Read Geo Data 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

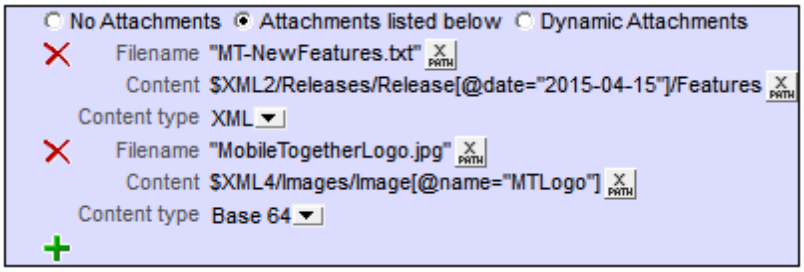
Files and images can be attached to the message. You can select one of three options for attachments:

- 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.



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** 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 and Share 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

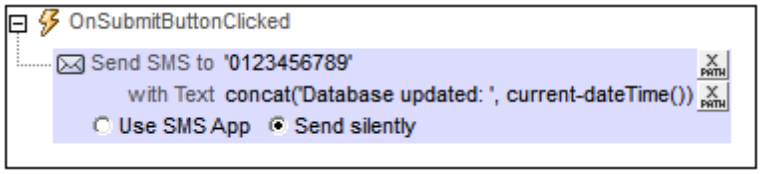
- `mt-email-attachment('MTNewFeatures.txt', $XML2/Releases/Release[@date='2015-04-15']/Features, 'XML')` returns `the Features node`
- `mt-email-attachment('MTLogo.jpg', $XML4/Images/Image[@name='MTLogo'], 'Base64')` returns `an image file`

Note: Attachments work with Android and iOS clients only.

The tutorial Sharing Geolocations shows how the Share action can be used.

Send SMS To

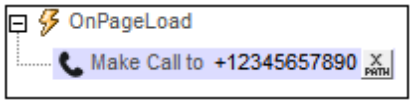
Sends an SMS to a specified number with a specified text. The receiver's number and the SMS text are specified using XPath expressions (*see screenshot below*). Numbers should not be enclosed in quotes. Numbers of multiple recipients must be separated by a semi-colon. A static global variable named **$MT_SMSAvailable** can be used to test whether SMS services are available on the client device. Variable values can be **true()** or **false()**.



You can choose whether to send the message via the SMS application on the mobile device or silently.

Make Call To

Makes a call to the number specified in the XPath expression of the definition (*see screenshot below*). Numbers should not be enclosed in quotes. A static global variable named **$MT_TelephonyAvailable** can be used to test whether telephony services are available on the client device. Variable values can be **true()** or **false()**.

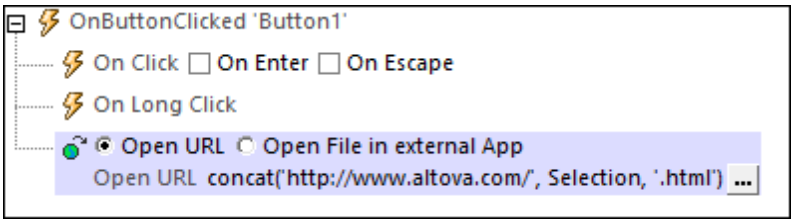


Open URL/File

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* radio button and then enter details of the URL or file to open.

Open URL

In the XPath expression dialog of the Open URL action, enter an expression that evaluates to a string that is the URL to be opened (*see screenshot below*).

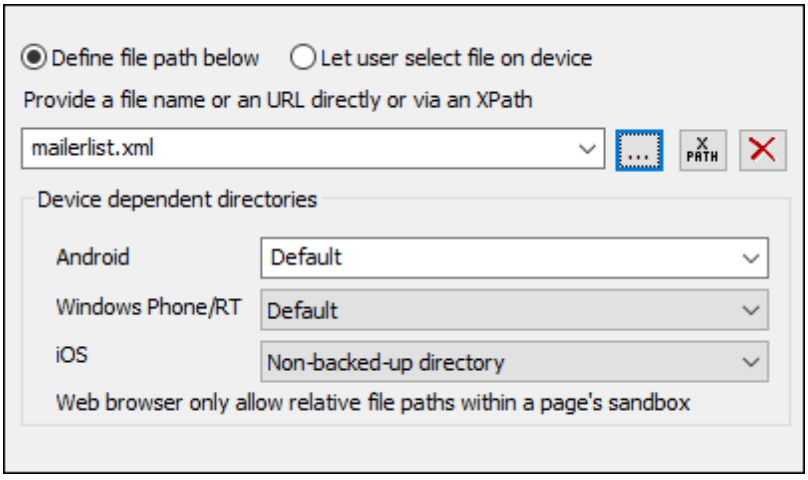


Open file

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 can select the file. 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

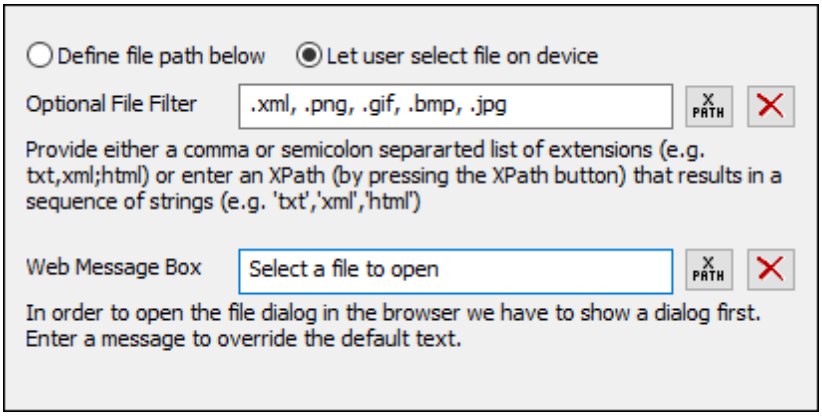
To directly specify (in the design) the device file to open, select *Define file path below* (*see next screenshot)*. Enter an absolute or relative file path, or an XPath expression that evaluates to such a file path. If you enter a relative path, then this path is resolved relative to the base directory you specify for that device type (*see list and screenshot below*).



- *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.



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:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

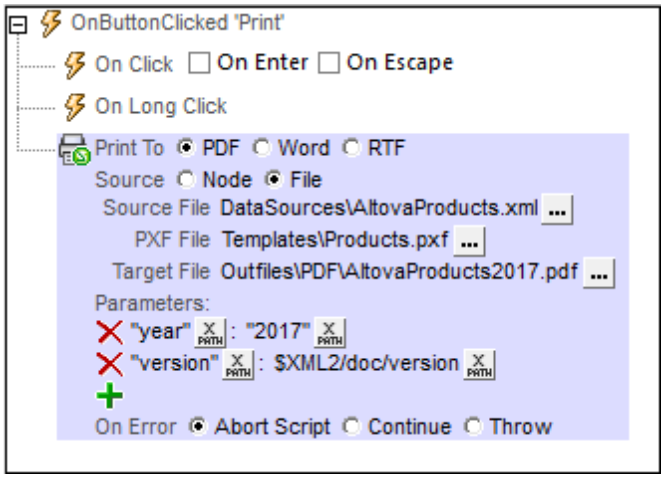
An error is registered only if the file does not exist. If the file exists, a success is registered—even if the file could not be opened by any device app.

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 data 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.



The Print To action takes the following options:

PDF/Word/RTF
Specifies the type of print output document: `.pdf`, `.docx`, or `.rtf`.

Source
You can select either: (i) an XML node of one of the design's data 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 a dialog appears in which you can browse the design's data source trees for 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.

Source File
If you choose *File* as your source, click the **Edit** button of the *Source File* option to select the source file. A dialog appears in which you can select a server file or client file. See the section *File locations* below for details of the available options. On selecting a file, you are asked whether the file should be deployed to the server or not. If the file is not deployed, it must be stored 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 so that it 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* 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 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 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:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 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.

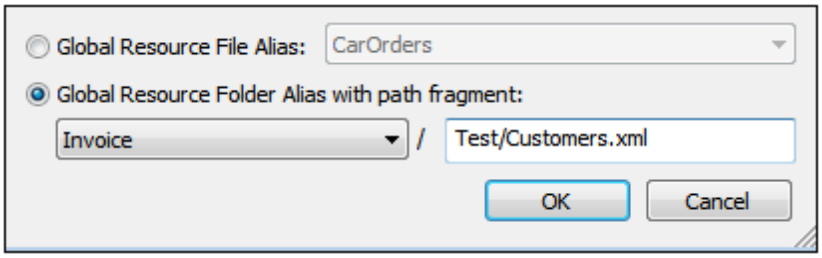
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design 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 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 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 for details. 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 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.

- *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). See the section Altova Global Resources for details.

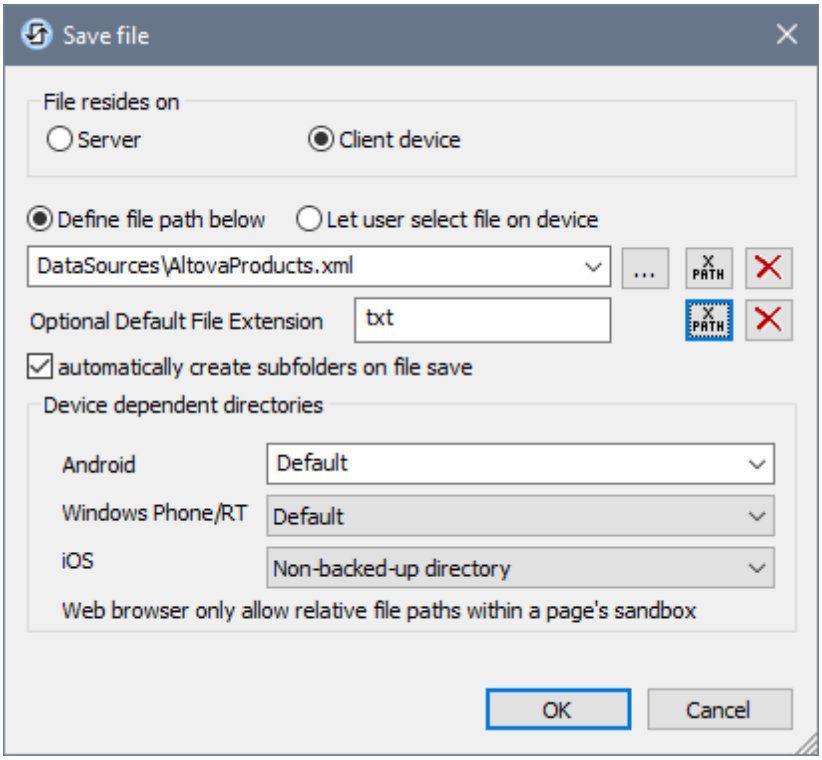
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



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**). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 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.



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 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 (*Source File* and *Target File* options), Load/Save File, Load/Save Image, and Load/Save Binary File.

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 (**Tools | Options**).

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.

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 data source.

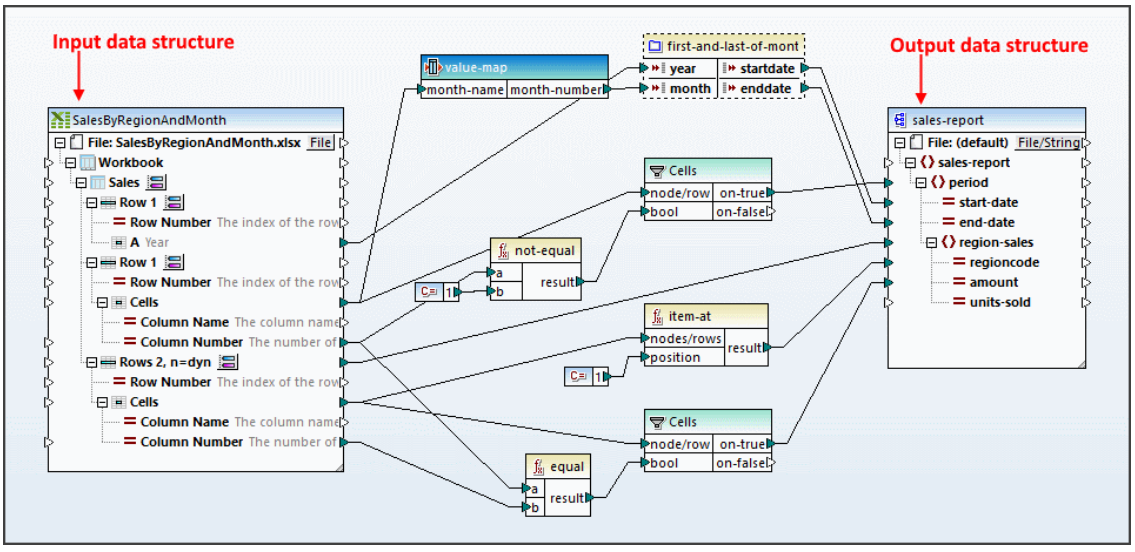
To carry out the MapForce transfer, you will need to use the following additional components:

- *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 or a trial run on client, 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.

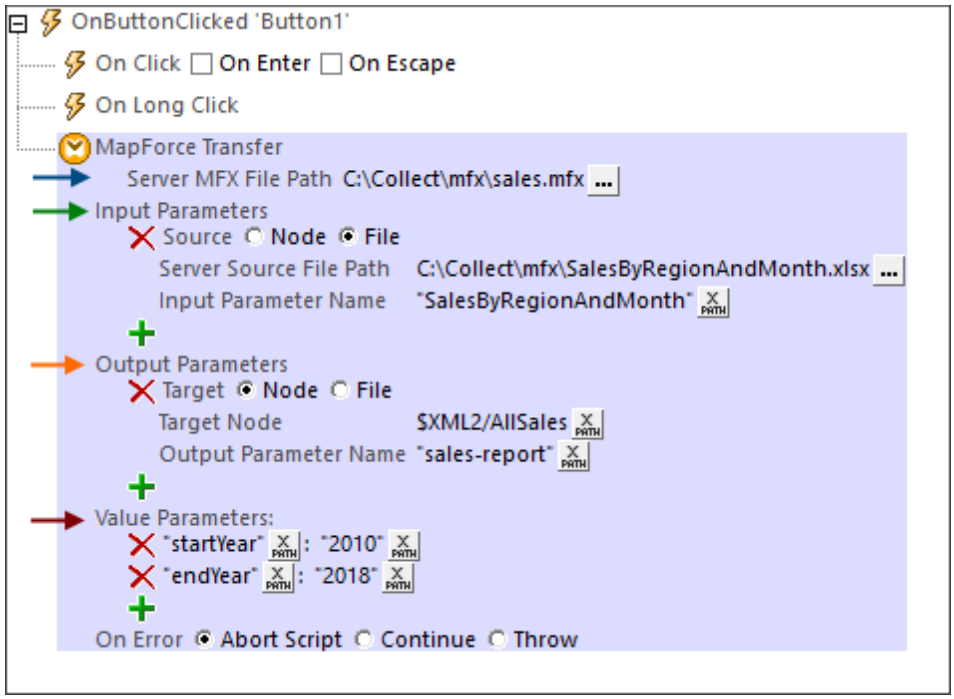
For more information about creating MapForce designs and how mapping works, see the MapForce user manual. Also see the MapForce Server user manual.

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.



- *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.

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.

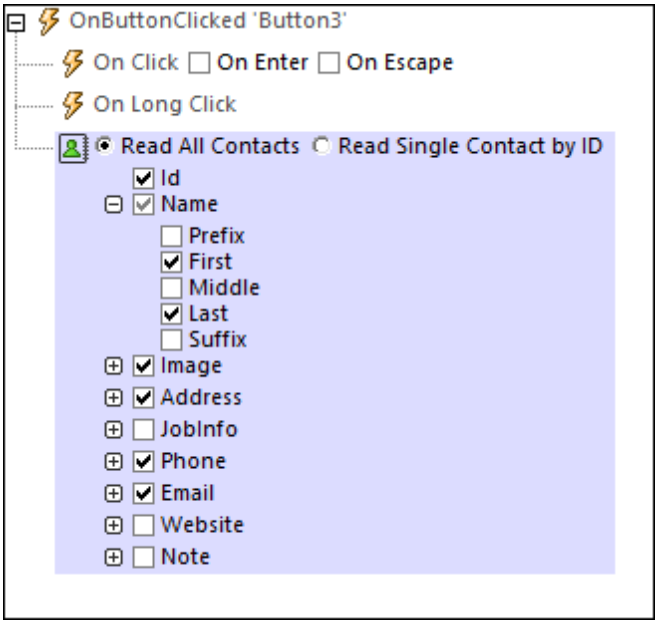
Use the wait-cursor action as follows:

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.

Read Contacts

When a Read Contacts action (*screenshot below*) is added to the design, the **$MT_CONTACTS** data source tree is automatically added to the design. When the action is triggered, contacts from the device's address book are read and stored in the **$MT_CONTACTS** tree.



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 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.

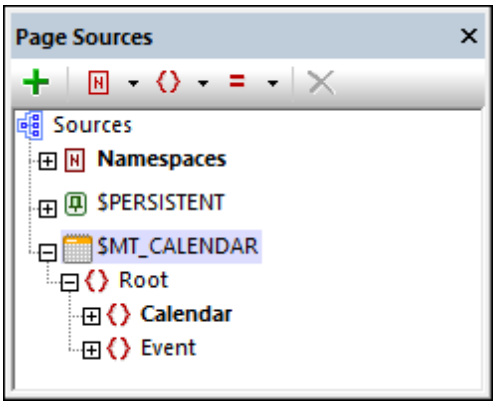
Access Calendar

When an Access Calendar action (*screenshot below*) is added to the design, the **$MT_CALENDAR** data source tree is automatically added to the design. At run time, depending on what kind of calendar action was selected, either (i) information from the device's calendars are read and stored in the **$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.

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.

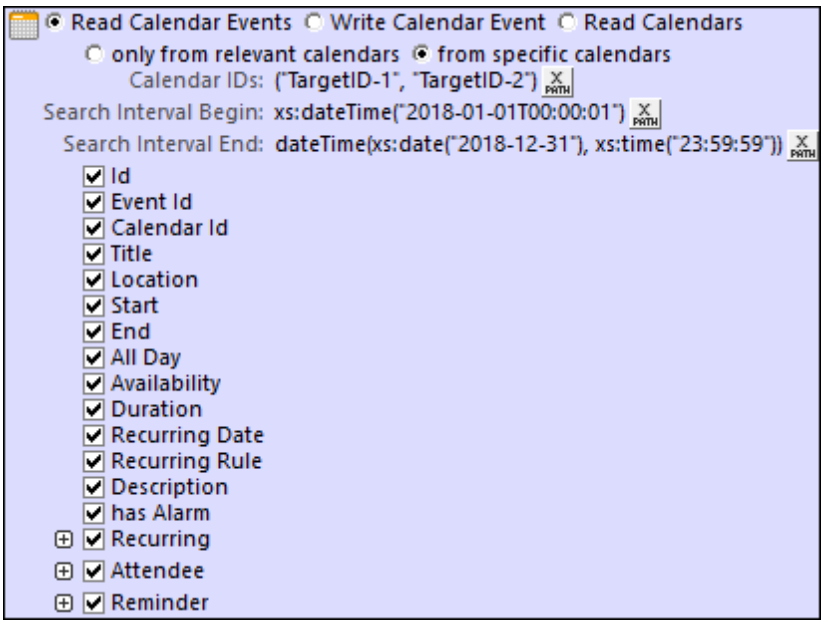


- *Write Calendar Event:* At run time, opens an event entry in the calendar app of the end user's device. This event entry will be filled with information that you entered in the action's settings. The end user can now edit and save the entry to the device's calendar.
- *Read Calendars:* Reads information 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 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.



You can select what data fields of the event to read:

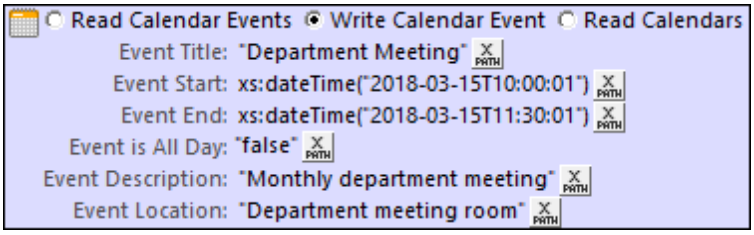
- *ID, Calendar ID:* A string that is the calendar's identifier.
- *Event ID:* A string that is the event's identifier.
- *Title:* The name of the event.
- *Location:* The venue of the event.
- *Start, End:* The start and end times of the event.
- *All Day:* Whether the event is to last all day. *All 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

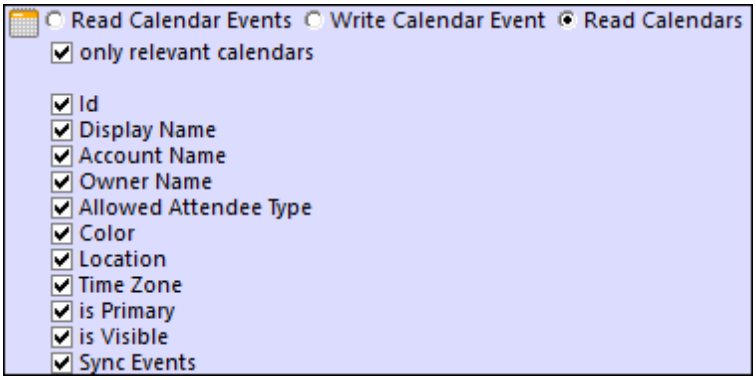
When this action (*see screenshot below*) is executed, it opens the device's calendar app and creates an event entry containing the data that you entered in the action. For example, an event entry created by the action shown in the screenshot below will contain the event's title, start and end times, description, and event location. Notice that in the screenshot below the values for *Event is all day* can be **`0`** (if false) or **`1`** (if true).



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.



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 (**Tools | Options**).

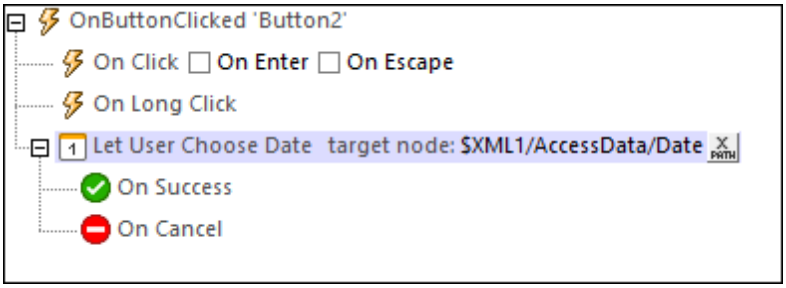
Structure of a sample calendar file to use for simulations

```
<?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>
```

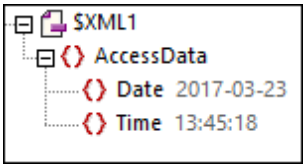
```
      <Event Id="1" Title="New Customer Lunch" Start="2018-05-14T12:30:00"
End="2018-05-14T14:00:00" Location="Sushi Restaurant">
         <Attendee Name="Alice" Status="Accepted" Type="Optional"
Relationship="Attendee"/>
      </Event>
   </Calendar>
   <Calendar Id="2" Name="Private">
      <Event 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>
```

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`.

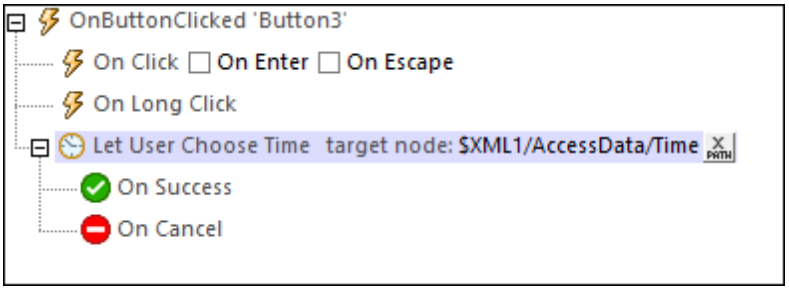


The screenshot below shows the relevant page-source node in a simulation, after a date has been selected on the client device.

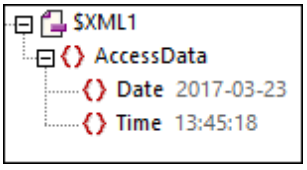


Let User Choose Time

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`**.



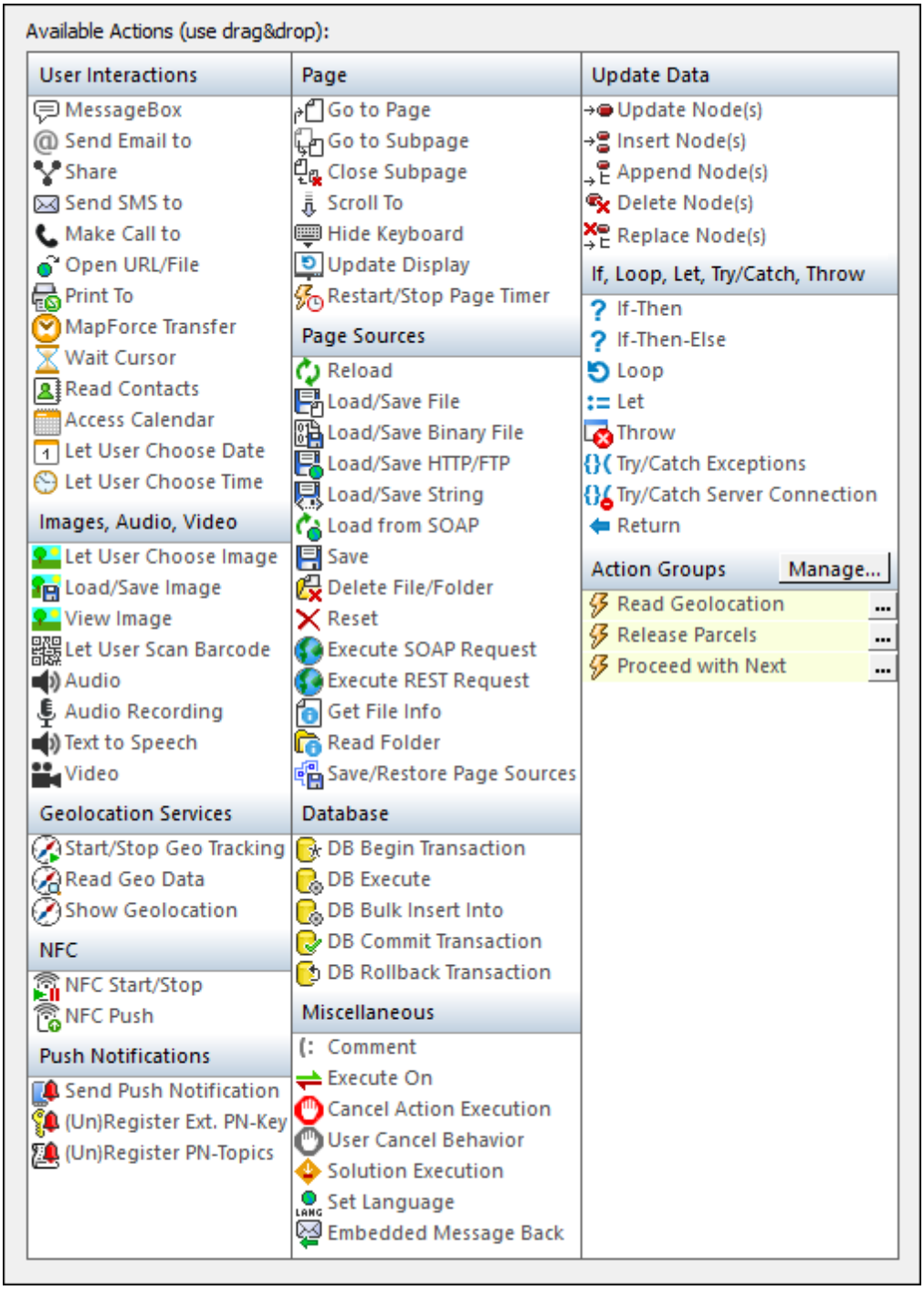
The screenshot below shows the relevant page-source node in a simulation, after a time has been selected on the client device.



10.2 Images, Audio, Video

The following actions are available in the Images group of the Actions dialog (*screenshot below*):

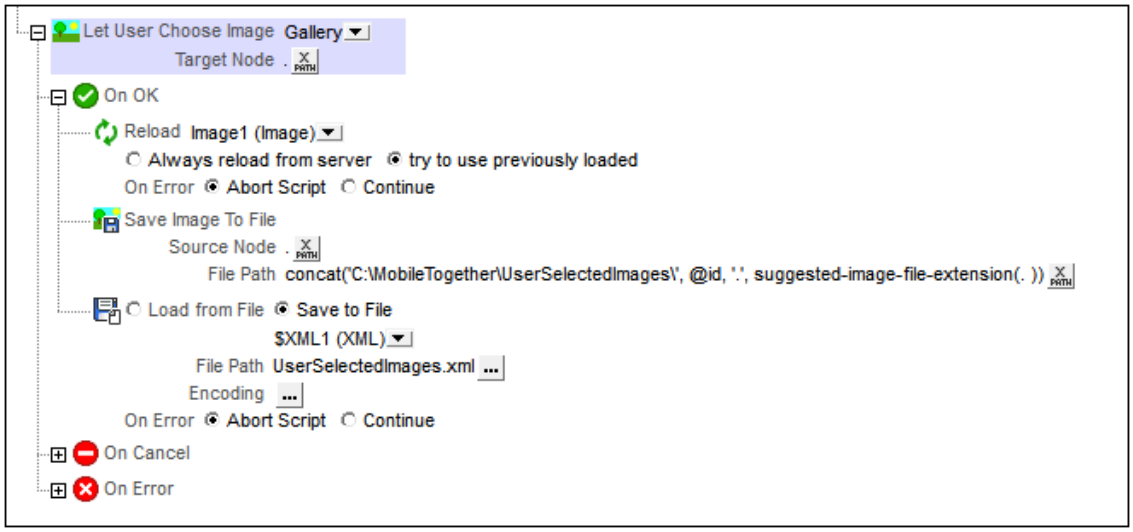
- Let User Choose Image
- Load/Save Image
- View Image
- ILet User Scan Barcode
- Audio
- Audio Recording
- Text to Speech
- Video



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 and Control Events*.

Let User Choose Image

The end user can select an image that can be saved to a node of the data 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.



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 data 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 the image control that displays the selected image; this updates the display with the selected image; (ii) Save Image to File if the image is required to be saved in as a binary image file (as opposed to being saved in an XML node as Base64-encoded text); (iii) Load/Save to File saves the XML data, including the newly added Base64-encoded image data to the page data 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.

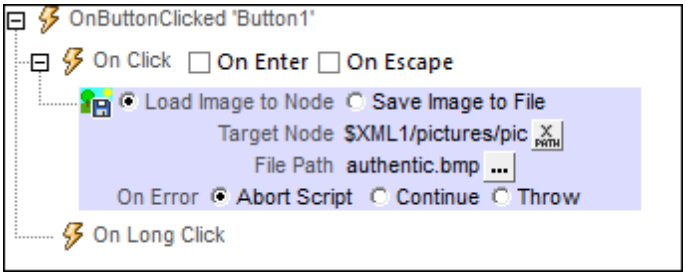
Load/Save Image

This action enables the following:

- Loading an image file into a data source node as a Base64-encoded image
- Saving a Base64-encoded image in a data source node as an image file at a server-side or other external location.

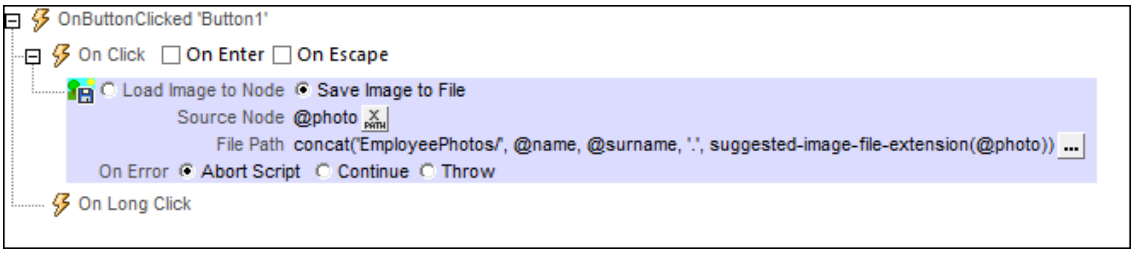
Loading an image file into a data source node

An image file can be loaded into a data 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 data 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.



Saving Base64-encoded image data as an image file

Base64 image data that is stored in a data 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 data 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).



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
```

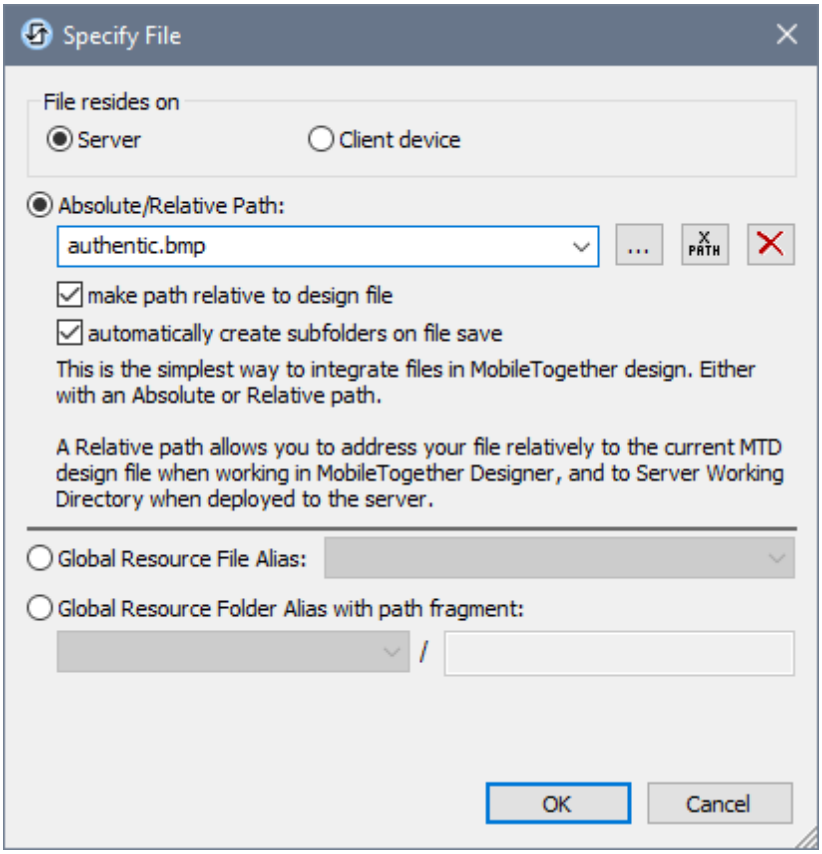
For an example of how to use this action, see the section Images Chosen by End User.

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 Specify 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 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.

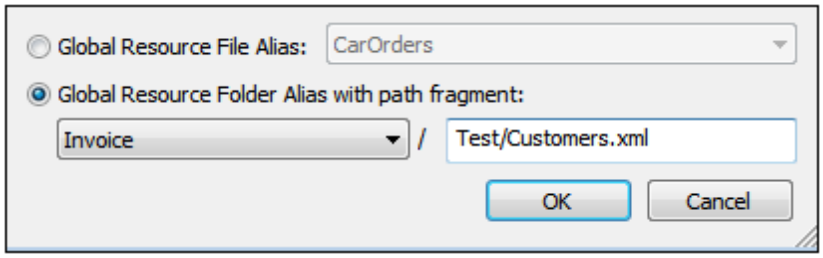


- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design 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 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 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 for details. 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 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.

- *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**). See the section Altova Global Resources for details.

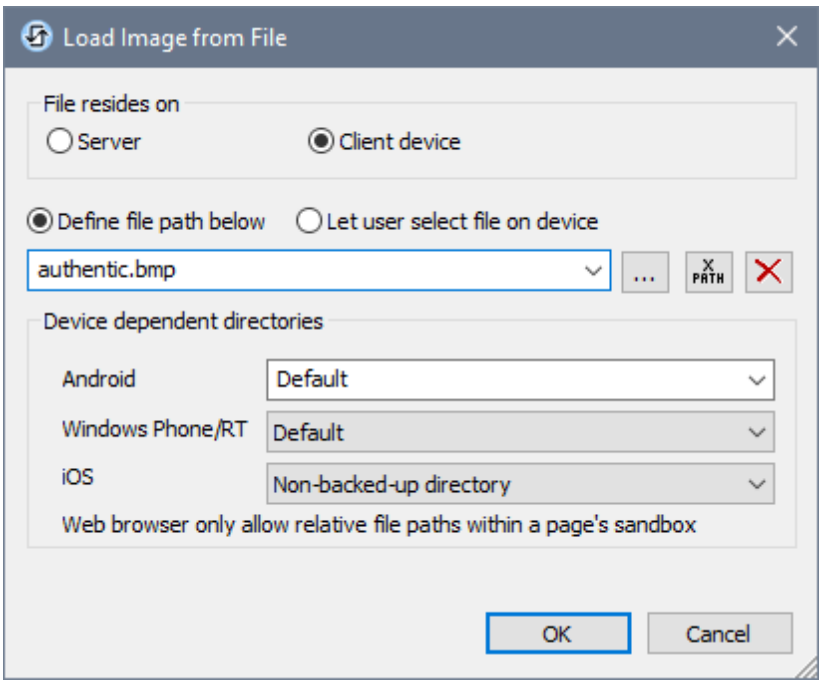
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



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**). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 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.



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 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 (*Source File* and *Target File* options), Load/Save File, Load/Save Image, and Load/Save Binary File.

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 (**Tools | Options**).

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.

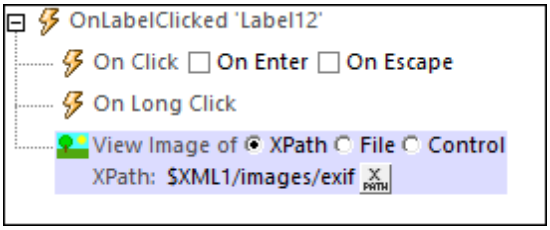
Error processing

The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

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.

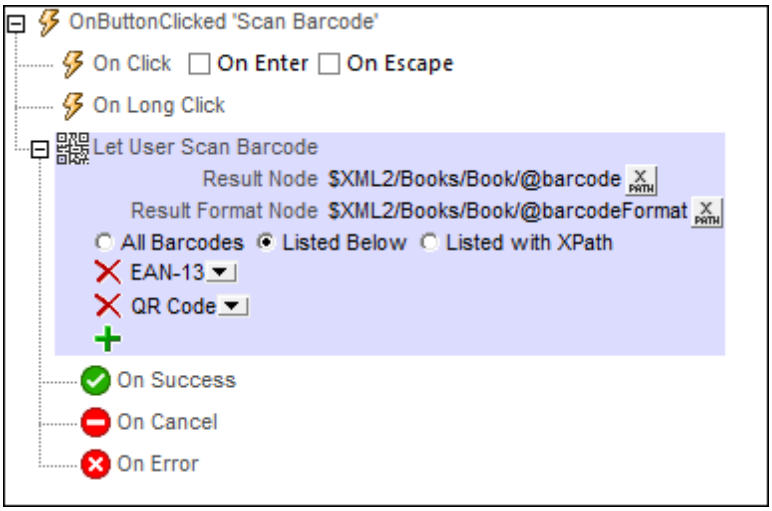


The following kinds of image selection are allowed:

- a Base64 image that is stored in an XML data source node
- 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
- via a control that displays the image; allowed controls are Image, Chart, and Signature Field 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. Select the control you want to display.

Let User Scan Barcode

When this action is triggered, the camera application of the client device is launched and the end user can scan a barcode. When the scan is completed, MobileTogether automatically enters the barcode and the corresponding barcode format into two separate XML nodes. For example, if an ISBN barcode is scanned, then the ISBN number and the ISBN barcode format (which is EAN-13) is saved to the two nodes specified in the definition of the action. This barcode information is then available to the design as XML data.



The Let User Scan Barcode action (*see screenshot above*) has the following options:

- *Result Node:* The data source node where the barcode data is saved.
- *Result Format Node:* The data 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:* 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.

Supported barcode formats

The currently supported barcode formats are listed below, together with limitations where these apply.

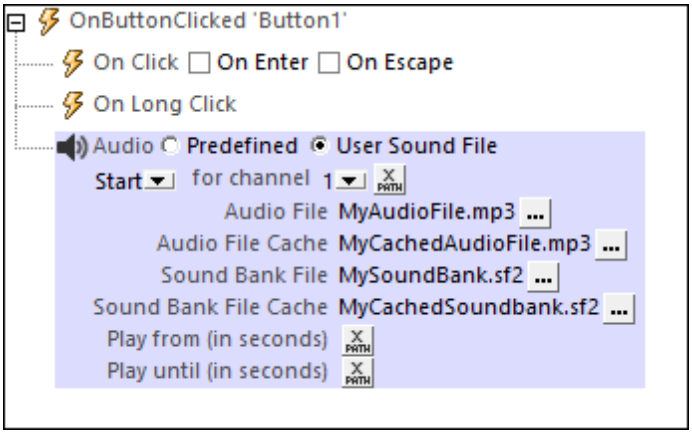
Format name	Notes
Aztec	
Codabar	*Not on iOS*
Code 39	

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*
RSS-Expanded	*Not on iOS*
UPC-A	*iOS: Detected as EAN 13*
UPC-E	

Note: No barcode support is available in web clients.

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.

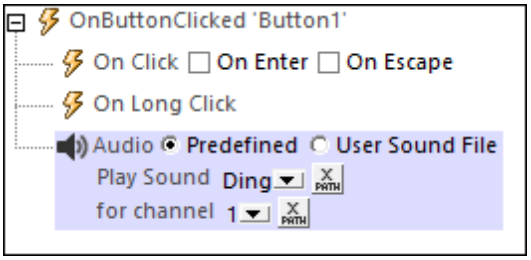


Predefined audio

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.



Using selected sound files

The Audio action also enables you 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.

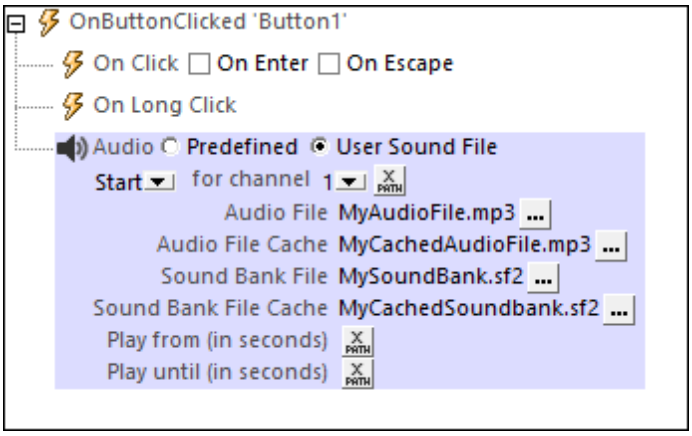
Audio can be played back on five channels (numbered **1** to **5**), and each Audio action is defined for one specific channel. So, for example, you could define the following sequence of actions:

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

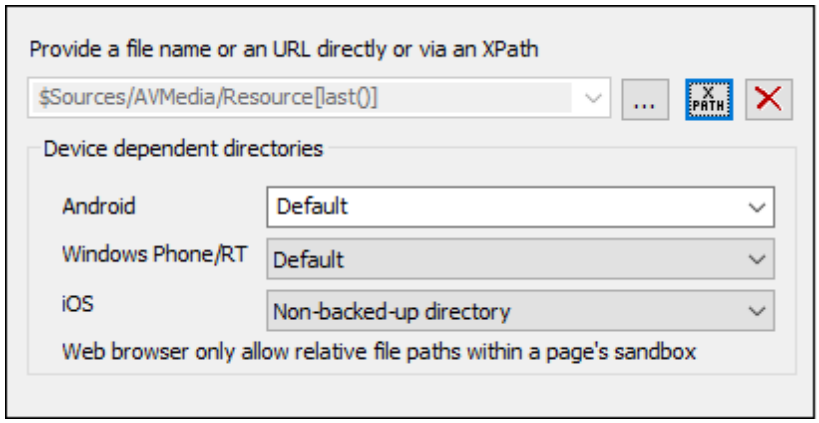
Typically, the Audio action will be defined on a control event such as a button click (*see screenshot below*). The event triggers the associated Audio action. The various Audio actions are described below. For an overview of using audio in solutions, see the section Audio Playback.

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.



- *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.



- *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. 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 action to load the binary audio/video data to a data source node; (ii) use the Save Binary action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 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, 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, 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.

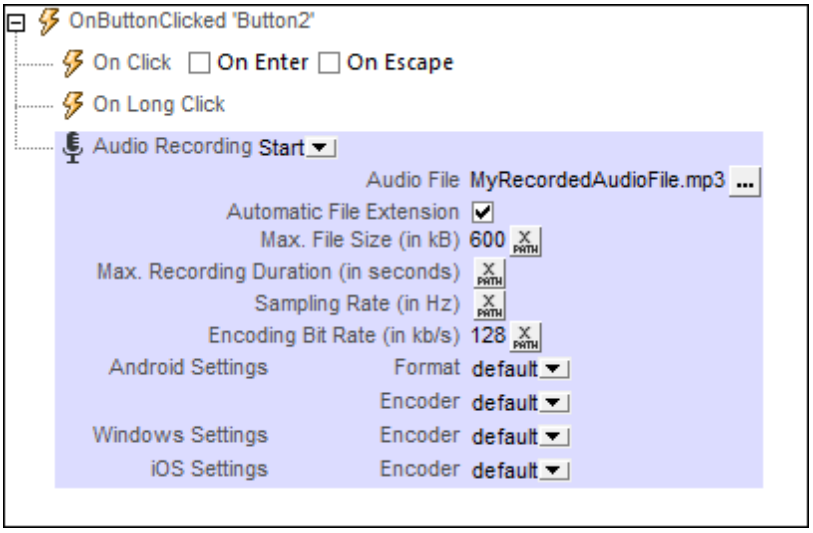
Additional points

The section Audio (Playback) provides an overview of the audio playback feature of MobileTogether. There, you will also find information about Audio (playback) events.

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.



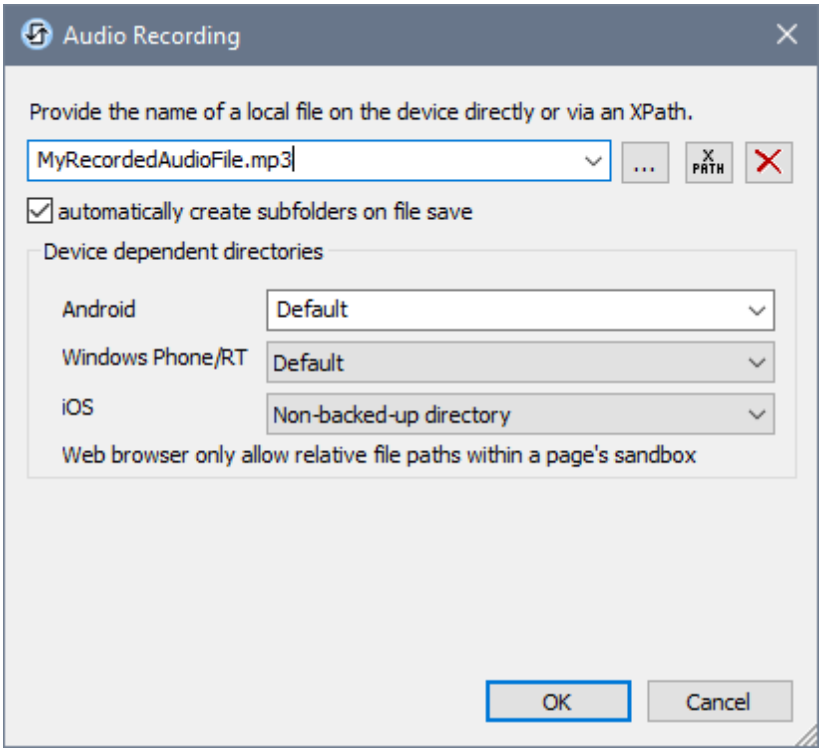
Note: If an audio recording is in progress when a solution is suspended, then the recording is stopped.

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.



- *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-last-file-extension` 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.

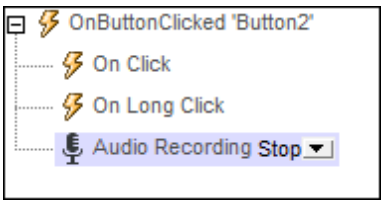
- *Encoding Bitrate (in kb/s):* Essentially the level of compression. When an audio file is compressed, details are removed. Usually these details are frequencies outside the human frequency range (20 Hz to 20 kHz) or frequencies that are reproducible only by high-end speakers. The tradeoff is between 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 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.

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.



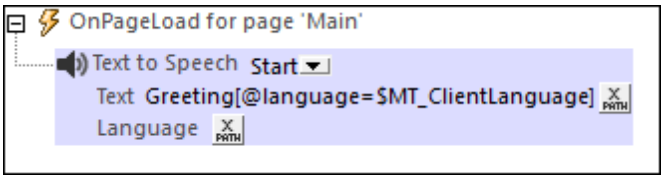
Note: A recording is also stopped when a solution is suspended or when the user leaves the page on which the recording was started.

Additional points

The section Audio Recording provides an overview of the audio recording feature of MobileTogether. There, you will also find information about Audio Recording events.

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 envision 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.



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.

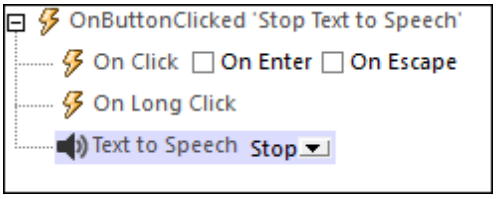
Note: Text to Speech playback is available on mobile devices only and cannot be simulated on MobileTogether Designer.

Additional Text to Speech actions

Text to Speech functionality can be extended by using the additional Text to Speech actions of the project properties of the design. For an overview of Text to Speech functionality, see the section Audio, Video | Text to Speech.

Stopping a Text to Speech action

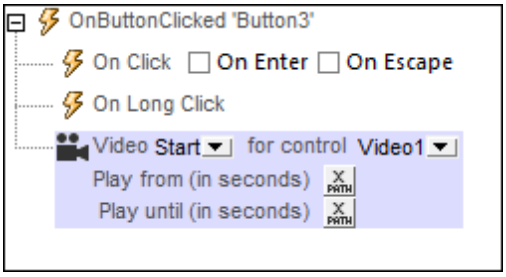
While a Text to Speech playback is running, it can be stopped by using the *Stop* function of the Text to Speech action (*see screenshot below*). When the action is triggered, any Text to Speech playback that is running at that time will be stopped. No specific text needs to be specified.



Note: Only one Text to Speech playback can be running at a time.

Video

The Video action (*screenshot below*) provides one of the following actions for a Video control on the current page: *Start, Pause, Resume, Stop, Seek (Jump) To*.



All the video controls on the current 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.

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.

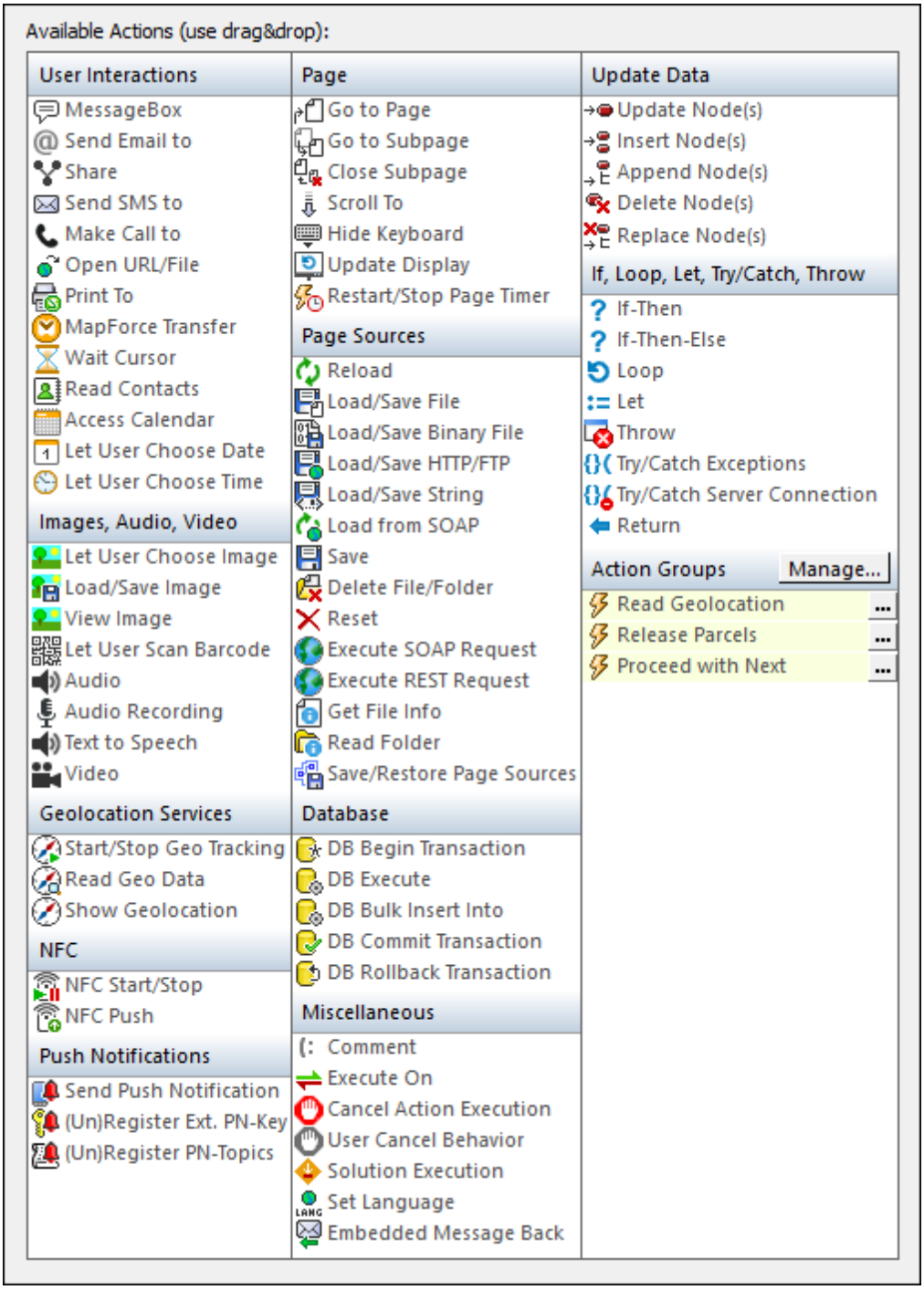
Note: If a Video control's **`Play on Load`** or **`Show Controls`** property is set to `false`, video playback can only be controlled via the actions of the Video action that have been listed above.

Note: Each Video control 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 context menu or by clicking the **Additional Dialog** button of the Video control's `Control Action` property.

10.3 Geolocation Services

The following actions are available in the Geolocation group of the Actions dialog (*screenshot below*):

- Start/Stop Geo Tracking
- Read Geo Data
- Show Geolocation

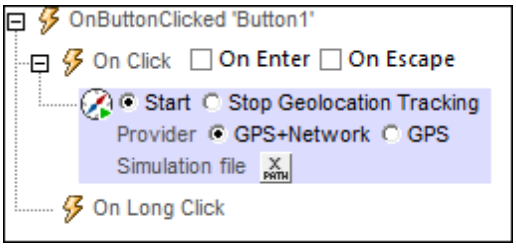


The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events and Control Events*.

The tutorial Sharing Geolocations shows how the Start/Stop Geo Tracking and the Read Geo Data actions can be used.

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 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.



The tutorial Sharing Geolocations shows how the Start/Stop Geo Tracking 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 data source that is best suited to the design:

- *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 data sources of the page when the Start/Stop Geo Tracking action or the Read Geo Data action is added to design.

Geolocations for simulations

For designer and server simulations, you can simulate a geolocation by specifying a geolocations XML file 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 is specified with this action, then the default geolocations file defined in the Geolocation Settings dialog is used.

Stop geo tracking

To stop geolocation tracking on the mobile device, select the action's *Stop* radio button (*see screenshot above*).

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 before this action is executed.

The **$MT_GEOLOCATION** tree is automatically added to the data sources of the page when the Start/Stop Geo Tracking action or the Read Geo Data action is added to design. The **$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

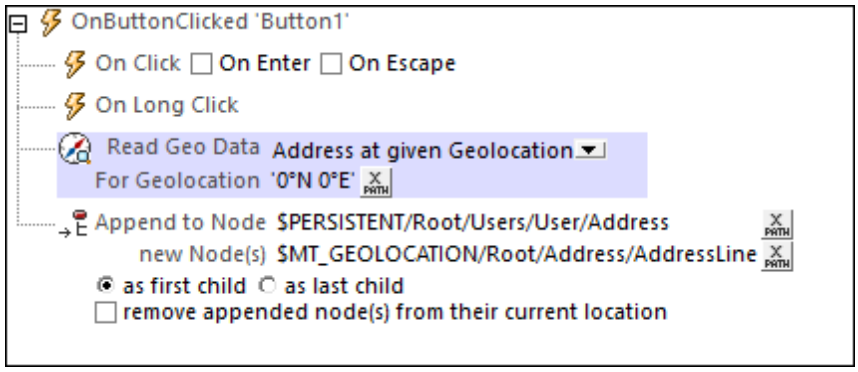
```
$MT_GEOLOCATION
<Root>
  <Location
    Provider=""
    Latitude=""
    Longitude=""
    Geolocation=""
    Altitude=""
    AccuracyVertical=""
    AccuracyHorizontal=""
    Speed=""
    Time=""
    MagneticHeading=""
  />
  <Address
    Locality=""
    SubLocality=""
    CountryName=""
    CountryCode=""
    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.



- *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, W/E)
  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/W**) 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, W/E)
  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/W**) is optional
  +/-D°M.MM' +/-D°M.MM'
  Example: **+33°55.55' -22°44.44'**

- Decimal degrees, with suffixed orientation (N/S, W/E)
  D.DDN/S D.DDW/E
  Example: **33.33N 22.22W**

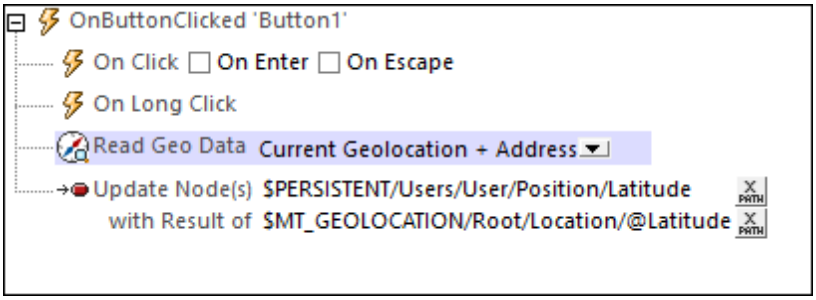
- Decimal degrees, with prefixed sign (+/-); the plus sign for (**N/W**) is optional
  +/-D.DD +/-D.DD
  Example: **33.33 -22.22**

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*).

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.



- *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.

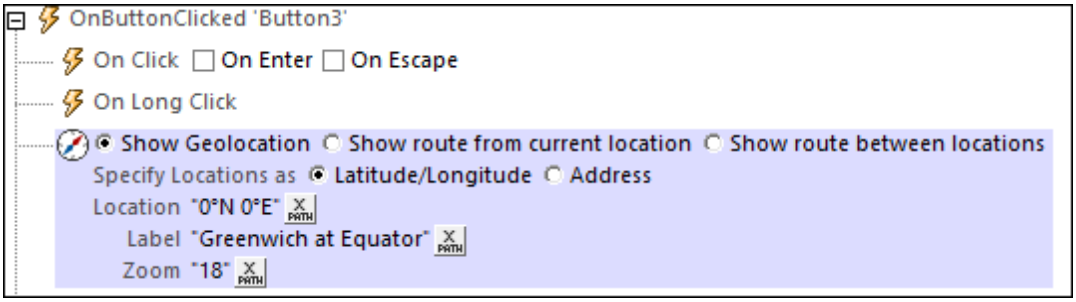
The tutorial Sharing Geolocations shows how the Read Geo Data action can be used.

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*):

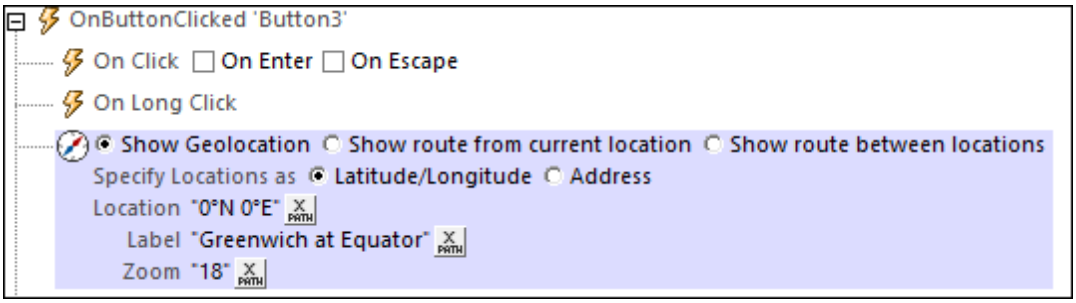
- *Show geolocation:* The location on the map of the coordinates that you specify
- *Show route from current location:* The route on the map from the current location to a specified location
- *Show route between locations:* The map route on the map between two specified locations

All locations in these options are given by their geolocation coordinates.



Show geolocation

The *Show Geolocation* action opens the map application of the mobile device, and places a pin and label at the specified geolocation.



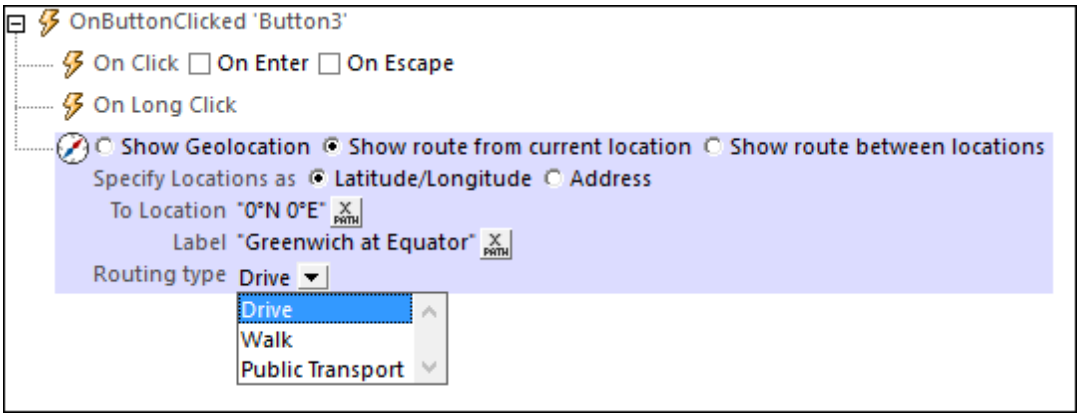
The *Show Geolocation* action has the following settings:

- *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.

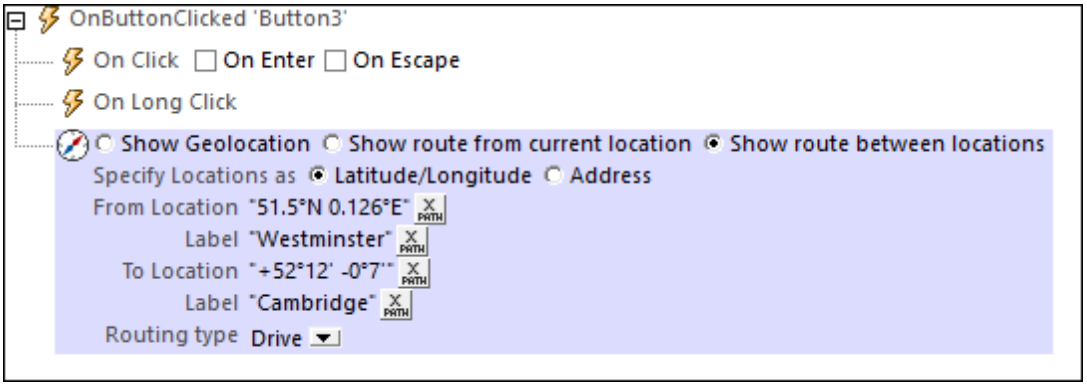


The *Show route from current location* action has the following settings:

- *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.



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`, `W/E`)
  `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/W**) 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, W/E)
  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/W**) is optional
  +/-D°M.MM' +/-D°M.MM'
  Example: **+33°55.55' -22°44.44'**

- Decimal degrees, with suffixed orientation (N/S, W/E)
  D.DDN/S D.DDW/E
  Example: **33.33N 22.22W**

- Decimal degrees, with prefixed sign (+/-); the plus sign for (**N/W**) is optional
  +/-D.DD +/-D.DD
  Example: **33.33 -22.22**

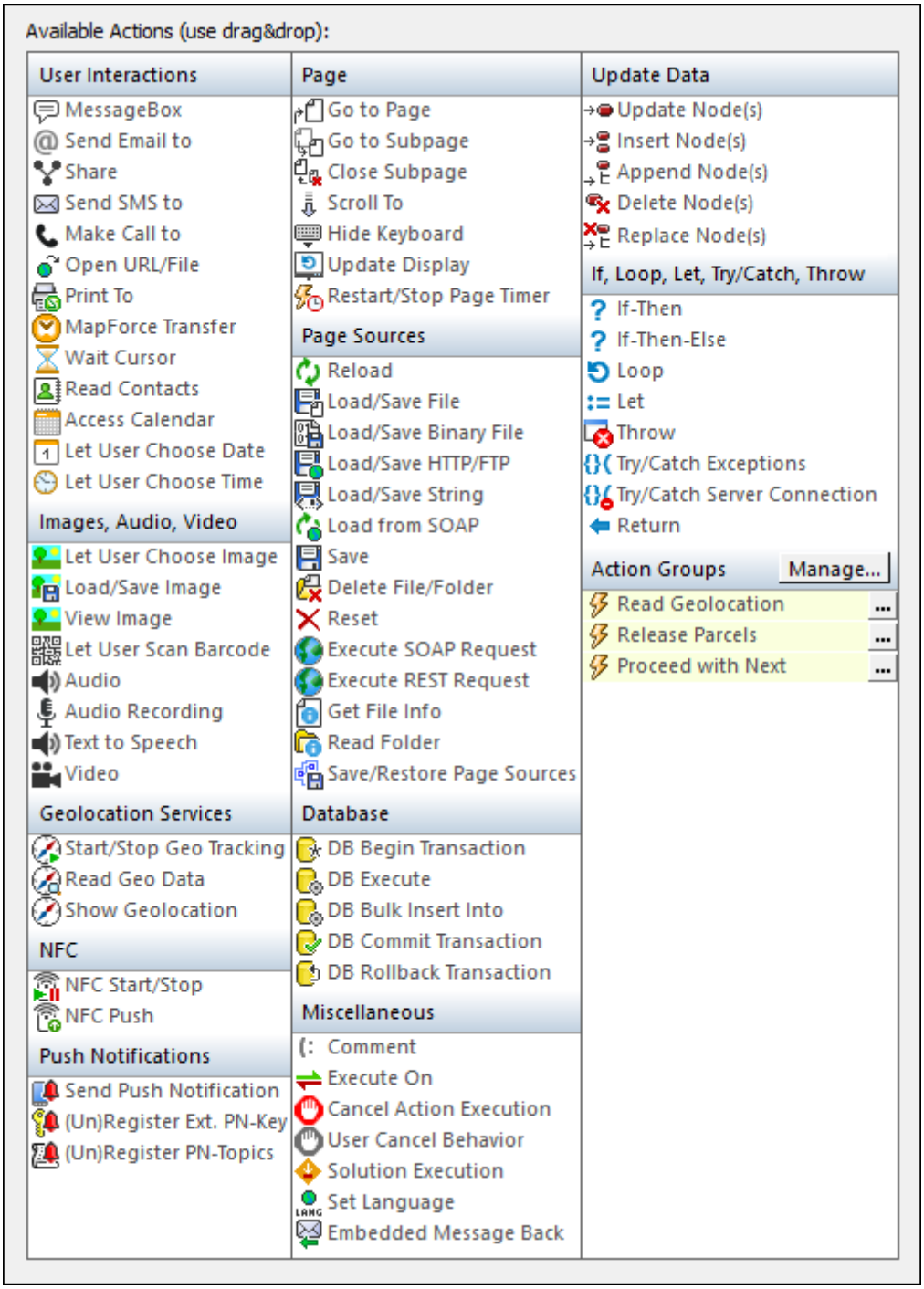
Examples of format-combinations:
33.33N -22°44'55.25"
33.33 22°44'55.25"W
33.33 22.45

For information about specifying geolocation data for designer and server simulations, see the section Geolocation Settings.

10.4 NFC

The following actions are available in the NFC group of the Actions dialog (*screenshot below*):

- NFC Start/Stop
- NFC Push

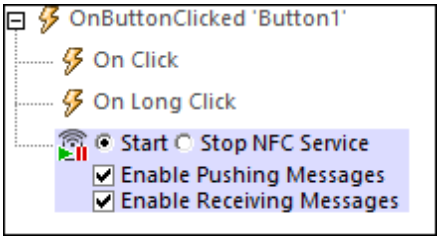


The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events and Control Events.*

See the section NFC for a broader description of NFC support in MobileTogether.

NFC Start/Stop

The NFC Start/Stop action (*screenshots below*) is used to start or stop the pushing and/or receiving of 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 (*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>
```

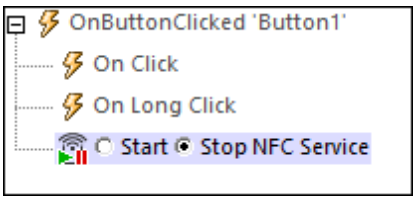
Pushing and/or receiving is started when the ***Start*** action is triggered. The sequence of steps that the action sets in motion is as follows:

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;

it is not automatically started.

4. Received data is stored in the **$MT_NFC** tree.

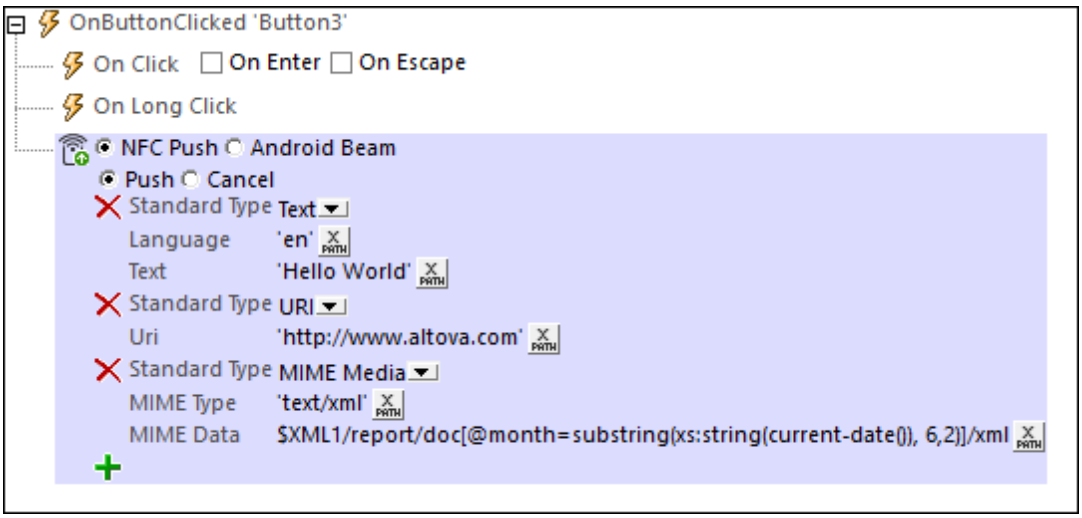
The ***Stop*** action stops the pushing and receiving of all messages.



To re-start the pushing and receiving of messages, trigger the *Start* action again.

NFC Push

The NFC Push action (*screenshot below*) defines the message to push or the file to beam. 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 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 can be sent multiple times (that is, to different devices). All that the end user needs to do is to place the receiving device within NFC range of the sending device. This continuous sending is stopped when NFC is stopped, or when an NFC Push is canceled (which is done by adding a new NFC Push action with the *Cancel* option selected; *see screenshot below*).



The NFC Push action provides the following options (*see screenshot*):

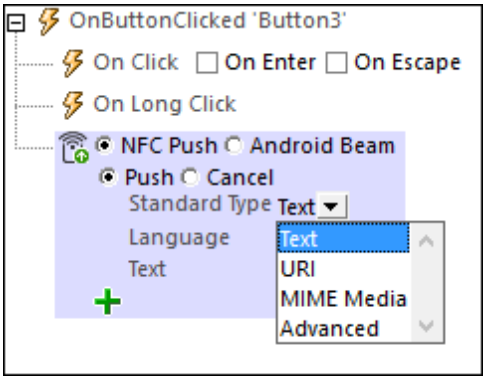
- *NFC Push:* to push a message
- *Android Beam:* to beam a file from one Android-Beam-enabled device to another Android-Beam-enabled device (available for Android devices only)

Select the radio button of the type of Push action you want to carry out (*see descriptions below*).

Note: Each action corresponds to one message/file 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.

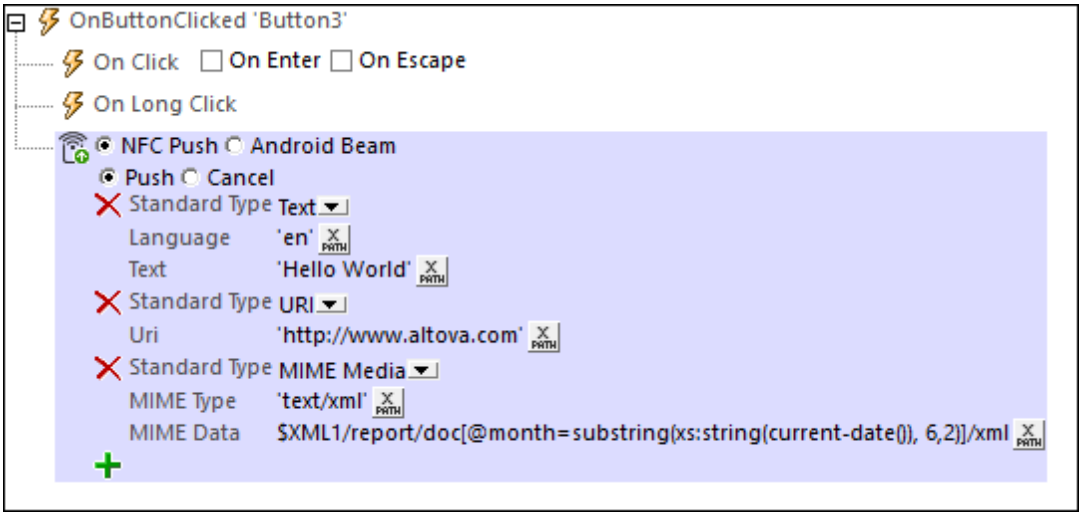
Pushing a message

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.



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 data source.



The following URI options are available:

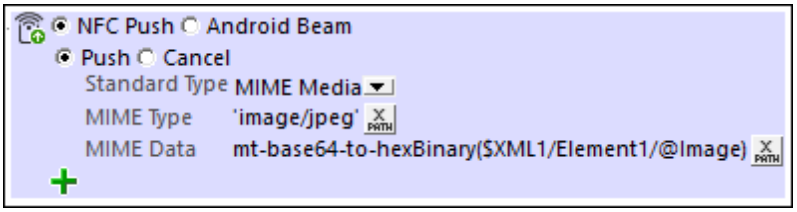
- *SMS URI:* The sent SMS will be opened in the receiving device's SMS app. The URI would look something like this: **`"sms:+439991234567?body=MyBody"`**
- *Telephone URI:* The telephone app on the receiving device will be opened, and the transmitted telephone number will be dialed. Example URI: **`"tel:+439991234567"`**
- *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. If you wish to convert a string to hexBinary, you can use the XPath extension function **`mt-string-to-hexBinary`**. Images can be converted from Base64 to hexBinary with the

mt-base64-to-hexBinary function. When an NFC payload is received on a device (*see Discovering and Reading NFC Tags*), the hexBinary can be converted back to a string or a Base64 image by using, respectively, the XPath extension functions **mt-hexBinary-to-string** and **mt-hexBinary-to-base64**.

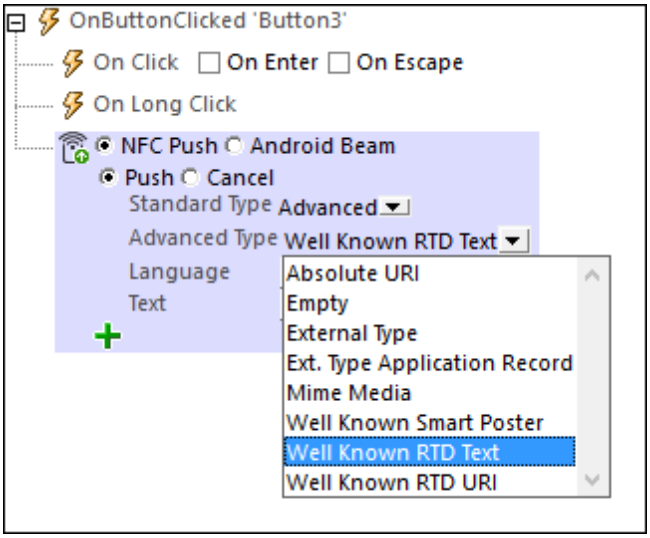
Pushing images

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 to hexBinary so that the image can be transported in the payload of the NFC message. For this conversion, use the XPath extension function **mt-base64-to-hexBinary**. To convert the hexBinary (on the receiving device) back to a Base64 image, use the function **mt-hexBinary-to-base64**.



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.

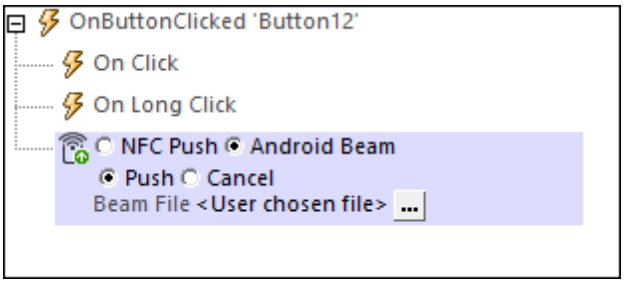


For a description of NFC's advanced types, see the NFC technical specifications.

Beaming a file with Android Beam

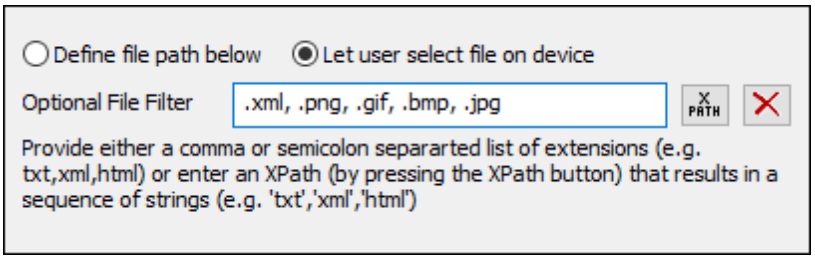
If Android Beam is enabled on two Android devices, then files can be beamed from one device to the other. Select the *Android Beam* and *Push* radio buttons (*see screenshot below*), and then

define how the file that the user will transmit is to be selected.



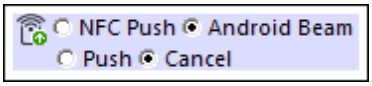
This file must be located on the client device. The file can be specified directly in the design or you can let the user select the file (*see screenshot below*).

If you specify the file directly and if the path is relative, then the path will be resolved relative to the base directory that you specify for Android devices. The screenshot below shows the dialog with the option *Let user select file on device* selected. The *Optional File Filter* setting filters the file types that can be opened via the browse dialog of the client device; 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'`).



Canceling a push/beam

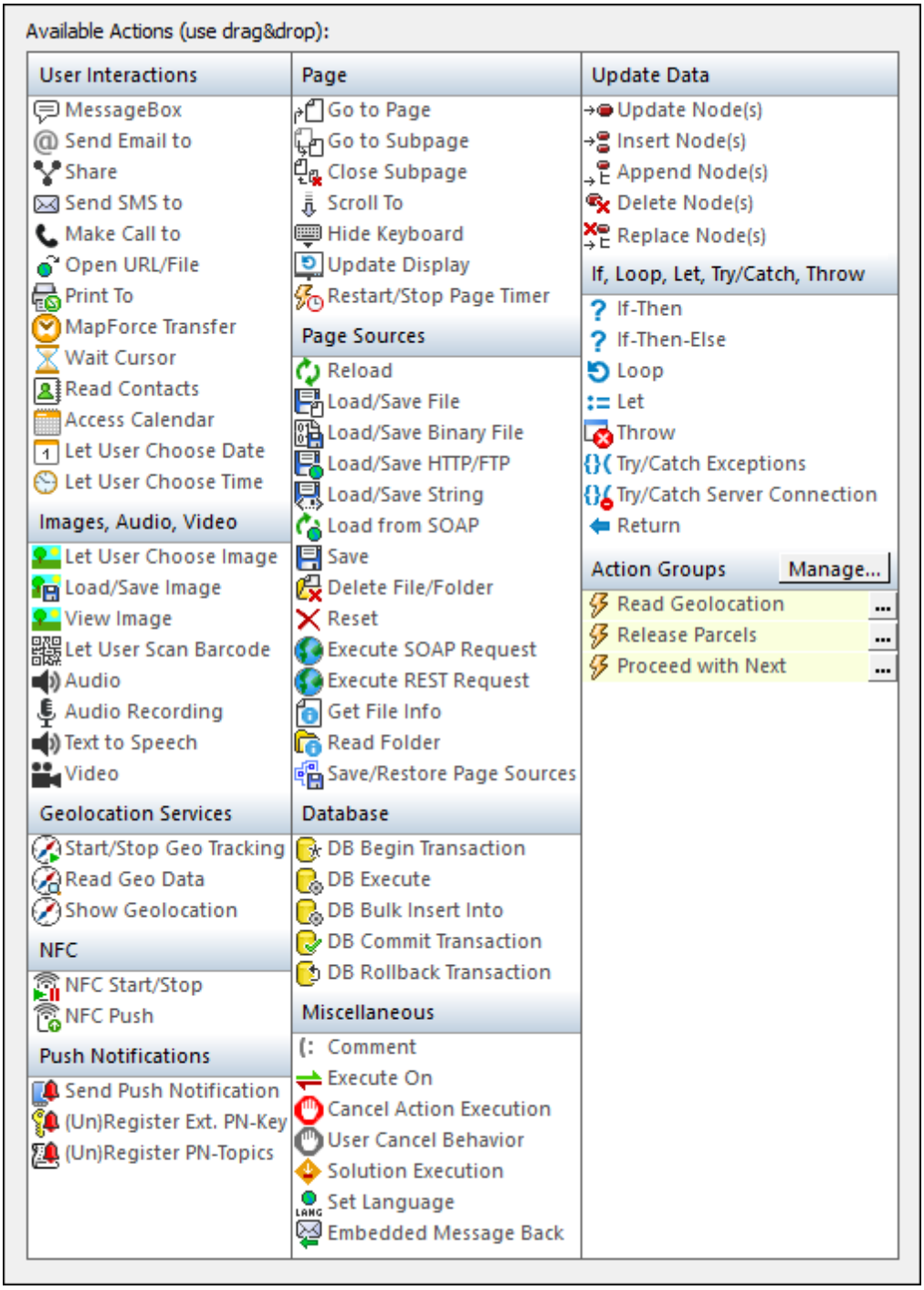
The Cancel action enables you to cancel the selected push or beam action.



10.5 Push Notifications

The following actions are available in the Push Notifications group of the Actions dialog (*screenshot below*):

- Send Push Notification
- (Un)Register Ext PN-Key
- (Un)Register PN-Topics

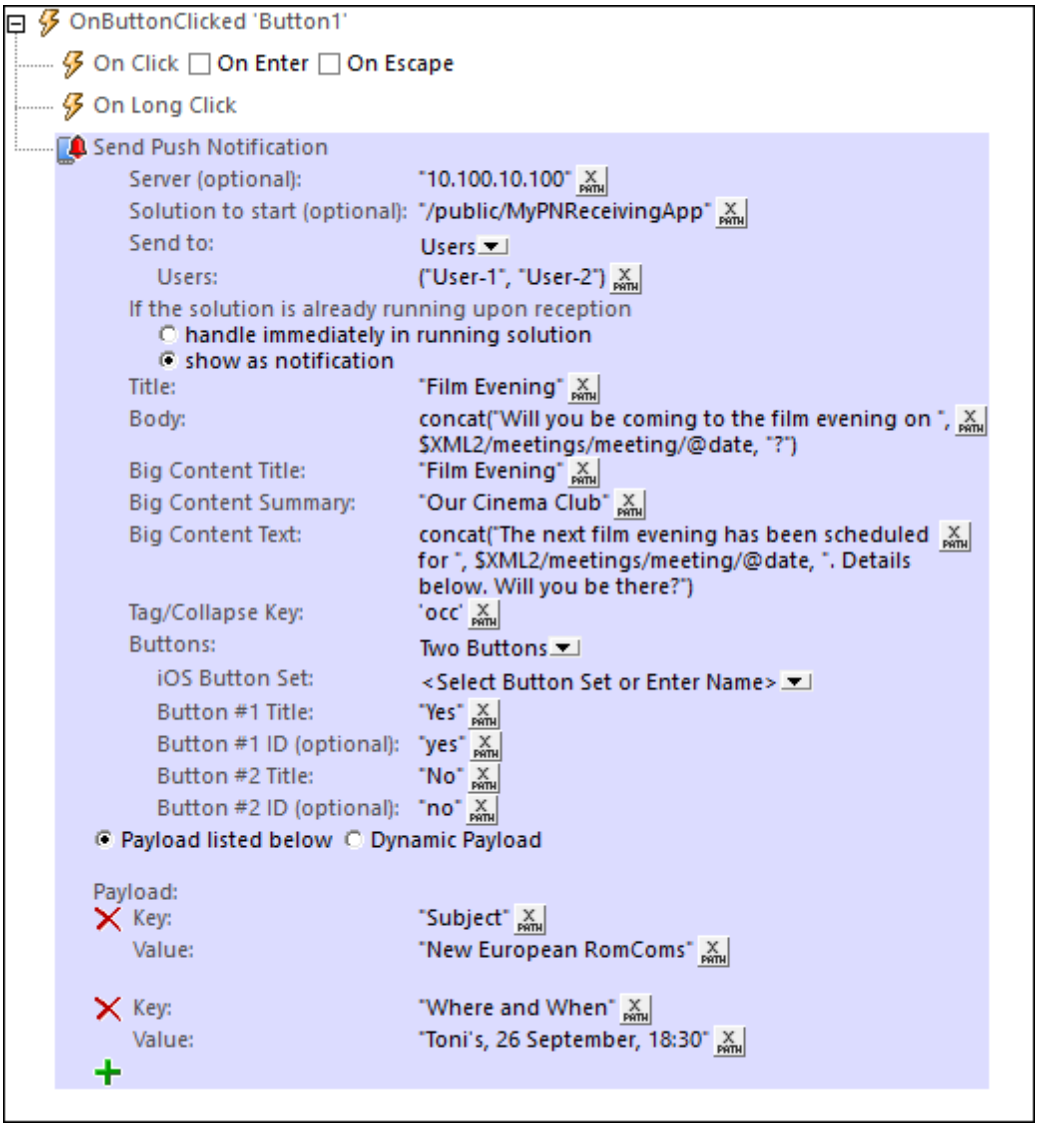


The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (*screenshot above*) is to right-click the page or control and select the page/control actions command. *See also Page Events and Control Events*.

See the section Push Notifications for an overview of how to use push notifications in MobileTogether.

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.



The action's settings, which are listed below, define the different parameters of the push notification:

- *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 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 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` 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` 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.

- *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 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** 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.

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 can be carried out according to which PN button the user has tapped. These conditional actions to carry out are defined in the **OnPushNotificationReceived** 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.

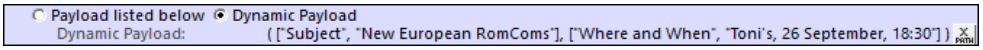
The payload is sent as an array of key–value pairs. Each key–value pair is placed (in index order) in an **Entry** element of the receiving solution's **$MT_PUSHNOTIFICATION** page source.

```
$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.

- 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.



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.

(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 a solution on that mobile device. Note that it is the device that is registered. If multiple solutions on a device register different external PN keys, then the device is registered with all these keys.

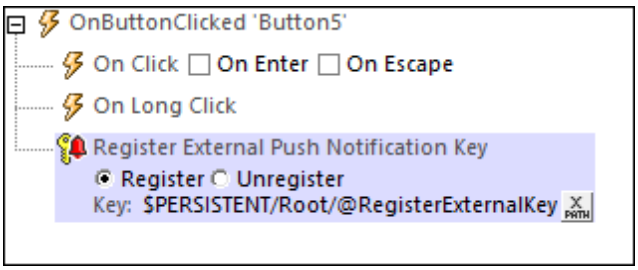
Among different ways to define a set of PN receivers when sending a PN, you can specify that the PN be sent to one or more external PN keys (*see Send Push Notification*). If a device is registered with one of the target external PN keys, then it will receive that PN.

Note: External PN keys can also be used in AppStore Apps.

See also: The Receiving Solution

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.



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 at any given time, you do not need to specify the key.

(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*). 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.

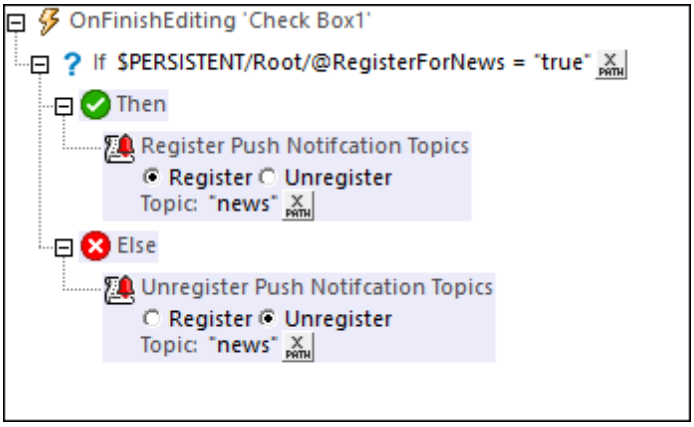
Among different ways to define a set of PN receivers when sending a PN, you can specify that the PN be sent to one or more specific topics (*see Send Push Notification*). If a device is registered for any one of these topics, then the device will receive that PN.

Note: PN topics are supported on Android and iOS; they are **not supported on Windows**.

See also: The Receiving Solution

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.



Unregistering

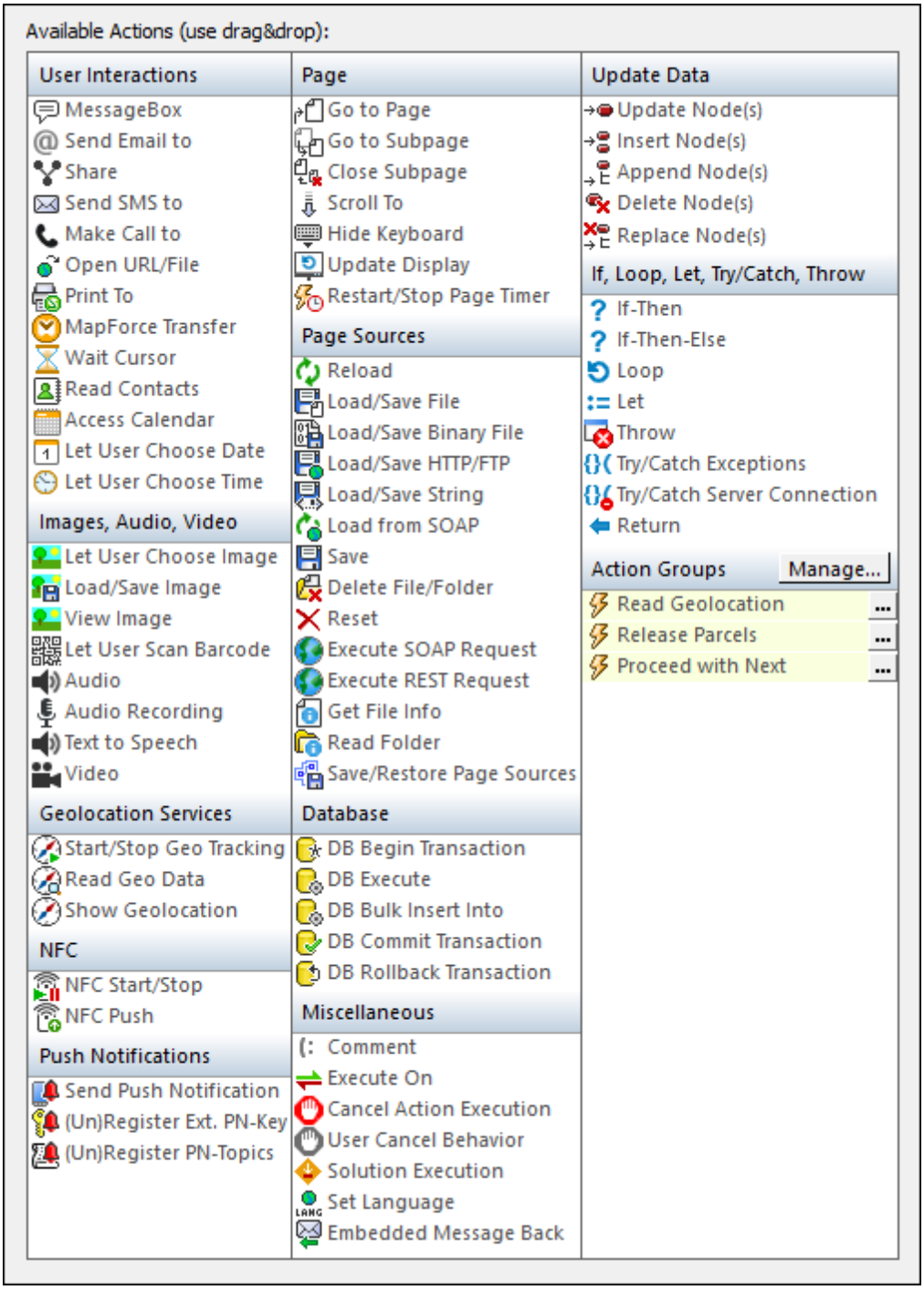
If at some point the user wishes to not receive PNs that are sent to a topic for which the device is registered, then the *Unregister* action can be used to cancel the registration. This could be done, for example, on a button tap.

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')`).

10.6 Page

The following actions are available in the Page group of the Actions dialog:

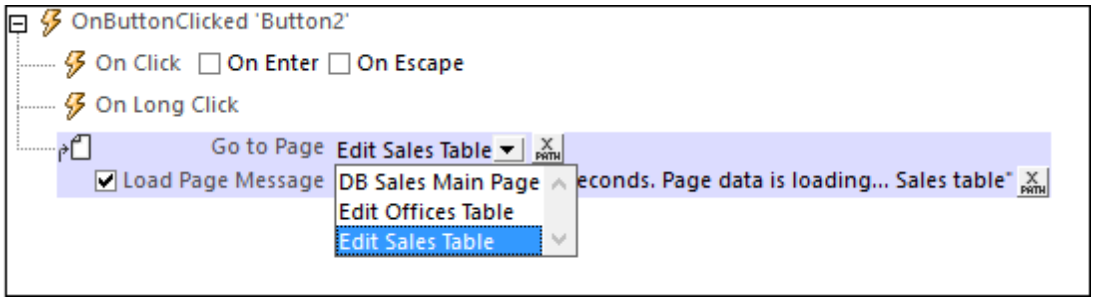
- Go to Page
- Go to Subpage
- Close Subpage
- Scroll To
- Hide Keyboard
- Update Display
- Restart/Stop Page Timer



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 and Control Events*.

Go to Page

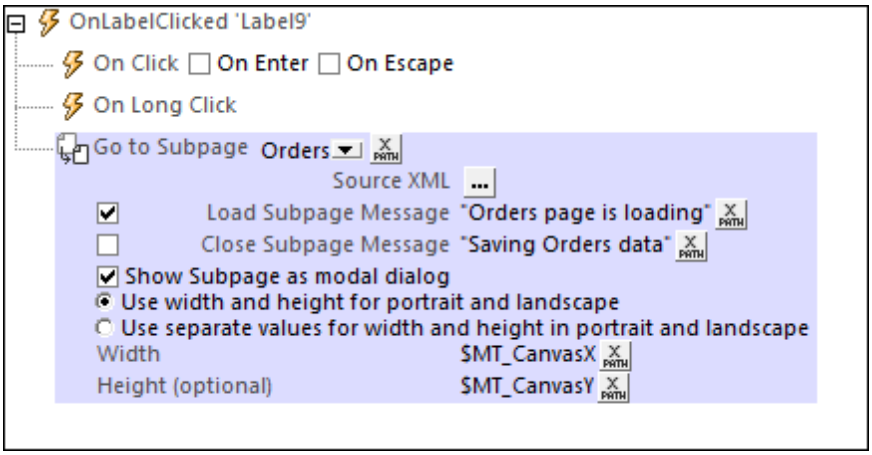
When triggered, the `GoToPage` action goes to the specified page. The page to go to is selected in the combo box of the action (*see screenshot below*) or via an XPath expression. If no other page exists in the project, no page will be available for selection in the combo box. In the screenshot below, 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).



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.

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"`.



The Go to Subpage action has the following settings:

- *Source XML:* Specifies whether and how subpage data should correspond to data 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" 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` 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).

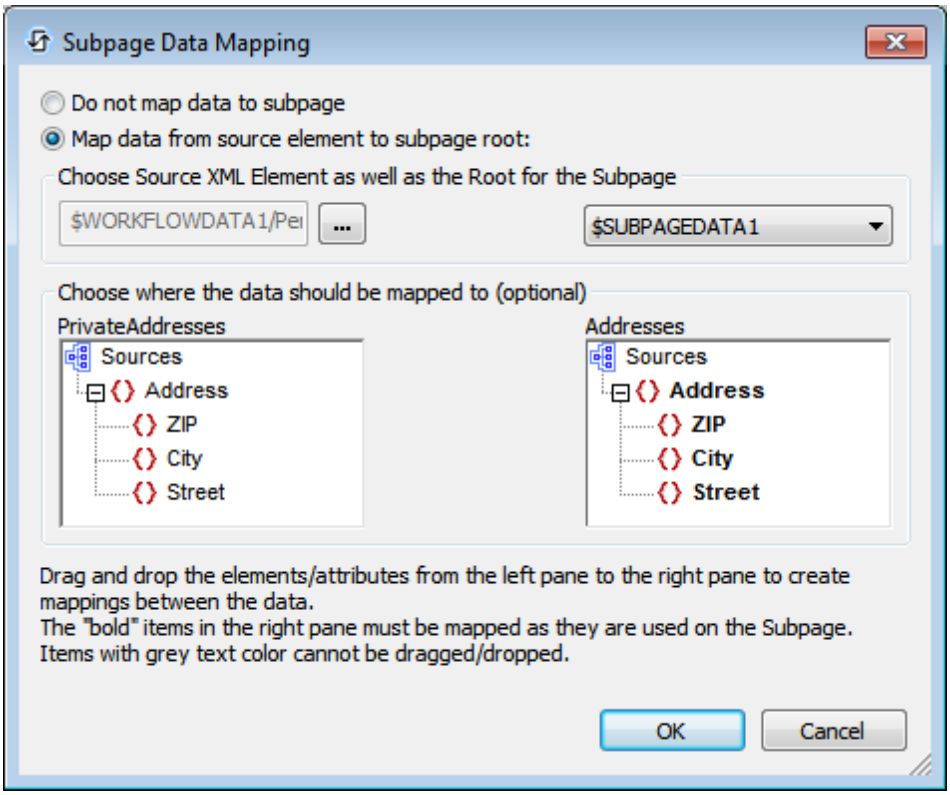
- In the screenshot above, the dimensions have been set to the device-specific width and height values (viewport dimensions) that are obtained dynamically from the device at run time.
- You can choose whether the modal dialog should: (i) have the same single set of dimensions for both portrait and landscape 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.

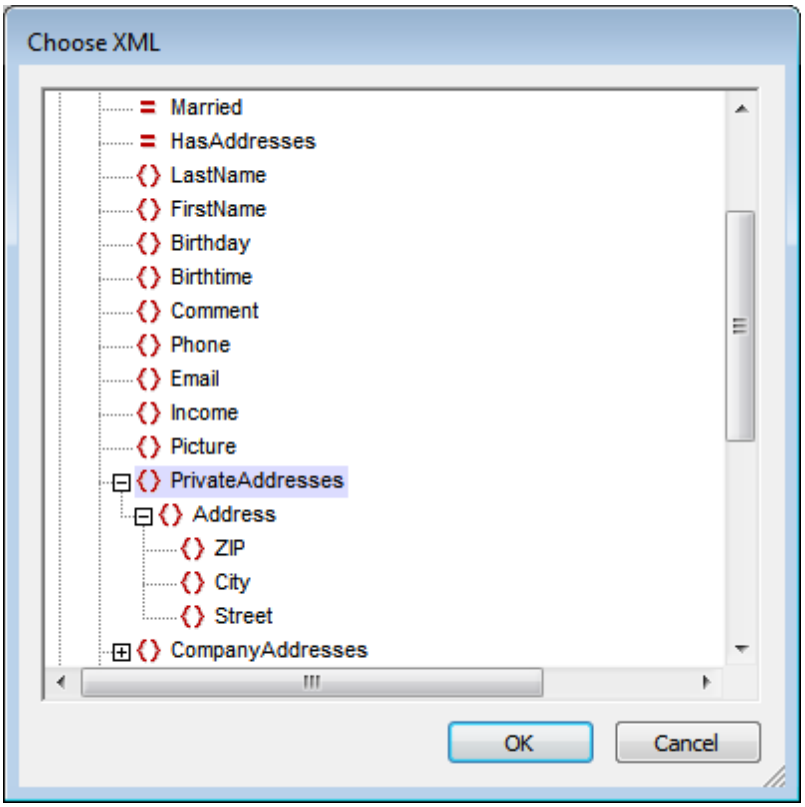
Source XML of subpages

Every subpage must have at least one data source in order for the subpage to be used effectively. Data in subpage sources 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, subpage data sources 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.



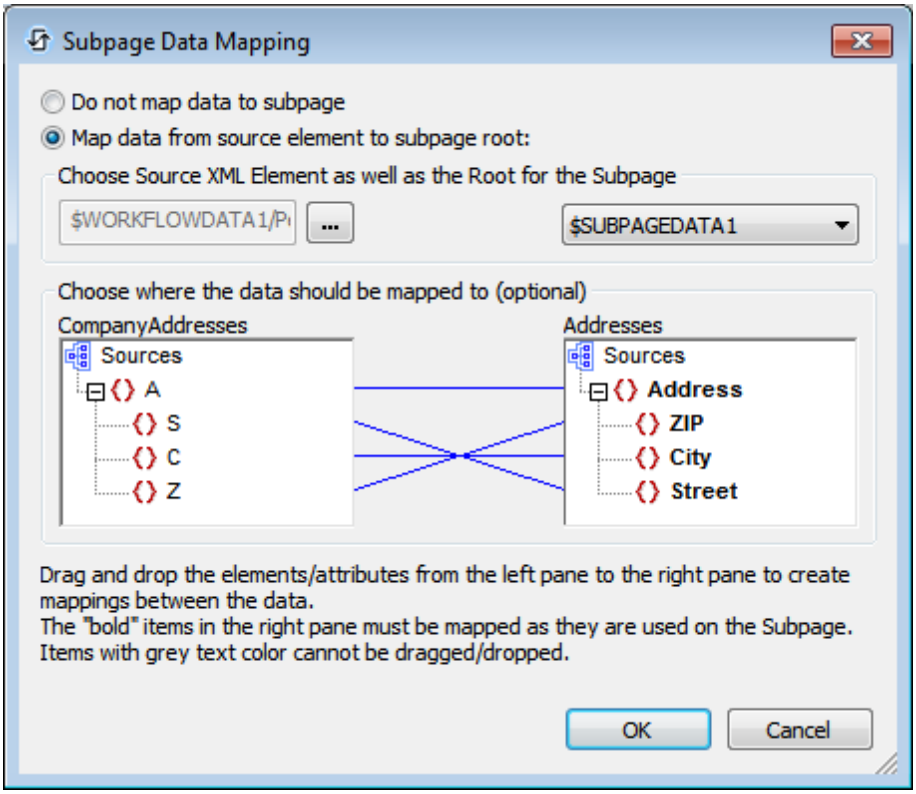


In the Subpage Data Mapping dialog (*screenshot above left*), the subpage data source is

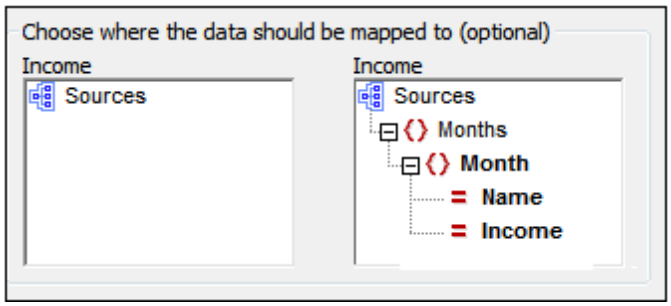
displayed in the right-hand source tree pane, while the top page data source is displayed in the left-hand source tree pane .

- Select the subpage data source from the combo box above the (right-hand) subpage pane. This combo box lists all the data sources of the subpage. Note that the subpage name is listed above the subpage pane.
- To select the top page data 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 data 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 data 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`).

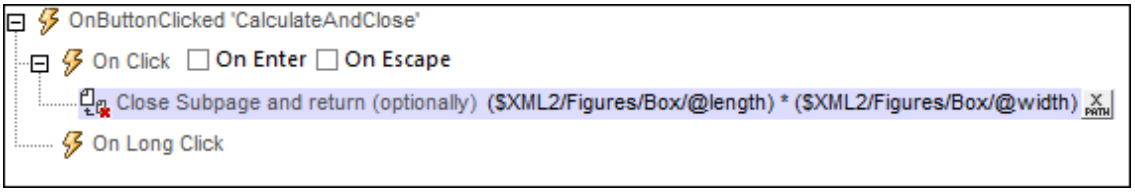


In the case described by the screenshot below, the content of the subpage data node `Income/Months/Month` is mapped to the top page data node `Income` (indicated by the title of the pane).


Choose where the data should be mapped to (optional)
Income
Sources
Income
Sources
Months
Month
Name
Income

Close Subpage

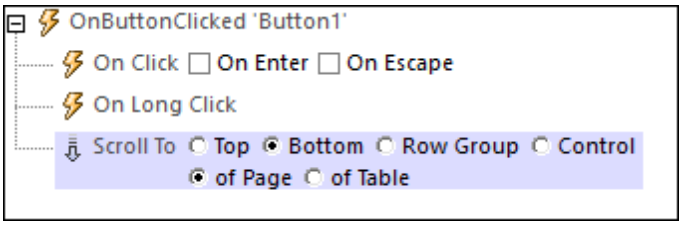
Closes the active subpage when triggered. Optionally, when the subpage is closed, a return value can be saved. The return value is generated by an XPath expression that you specify (*see screenshot below*). The value can be retrieved as the Subpage Result of the Let action.



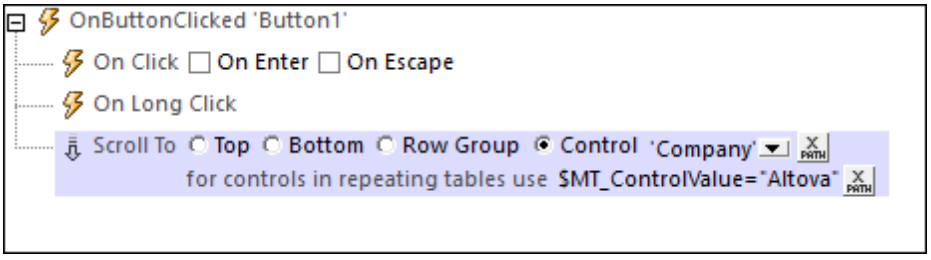
Scroll To

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*).



- Top or bottom of the selected table; only tables that have vertical scrolling 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 the first row group (of the selected table) for which the condition evaluates to **`true`**.
- 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, such as **`$MT_ControlValue`**, can be used to locate a control in a dynamically changing context.

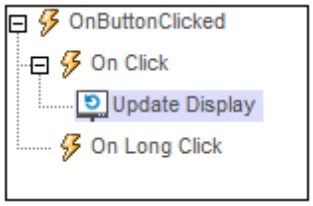


Hide Keyboard

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.

Update Display

Updates the display with the latest data when the action is triggered. In cases where a sequence of multiple actions are carried out when an action is triggered, the display is updated, by default, after the last action is carried out. The Update Display action enables the display to be updated as the actions for that event are executed. A good example where the Update Display action can be used is in the loop action. If the Update Display action is included within the loop, then the display is updated as the actions in each iteration of the loop are executed.



Restart/Stop Page Timer

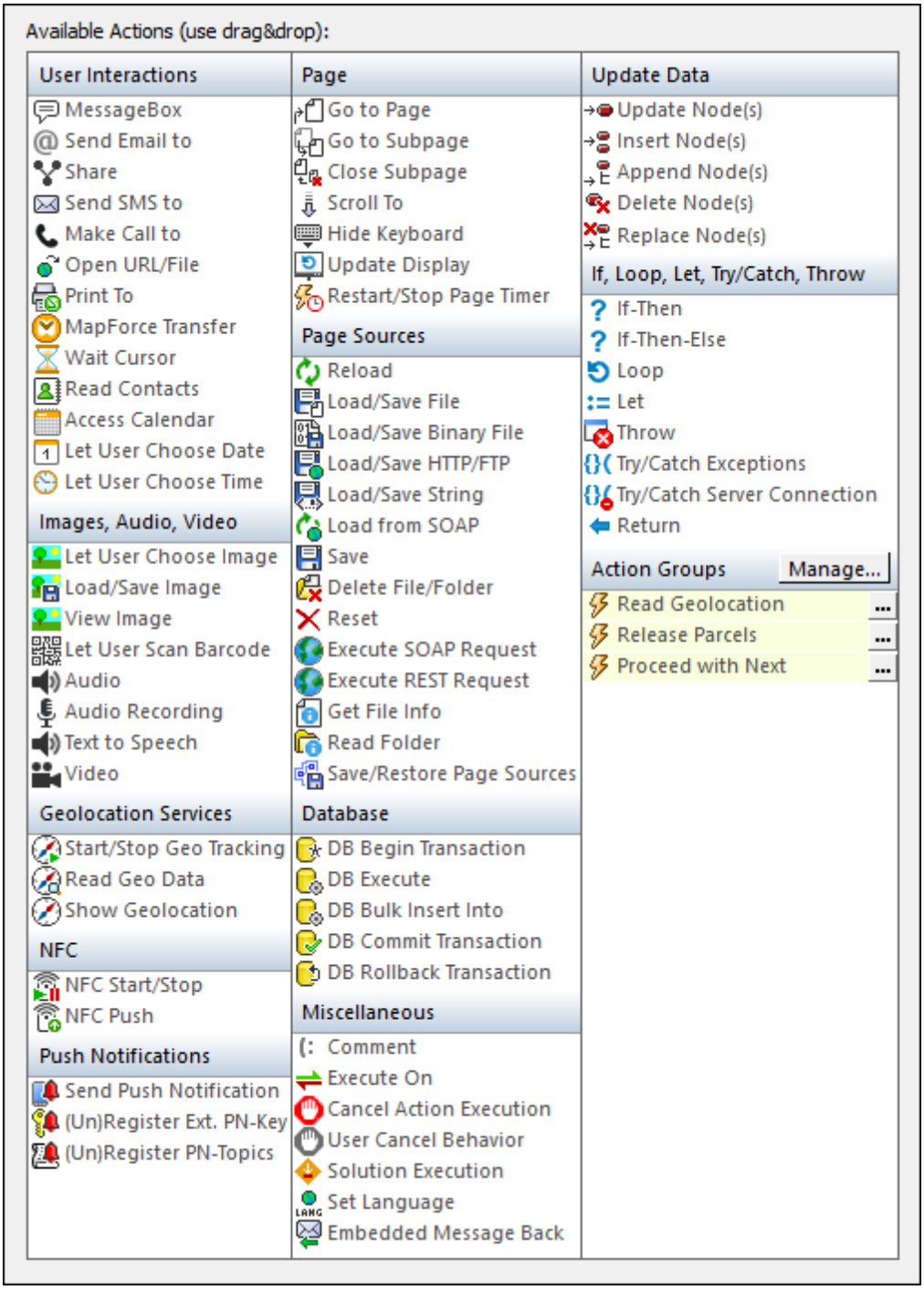
The page timer is configured in the **OnPageRefresh** event tab, and it is used to time the intervals between page refreshes. The timer's initial refresh interval is specified in the timer configuration. When the page is loaded, the timer is started, and the page is refreshed at the specified refresh 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 for an example*.
- *Stop Page Timer:* When a page is refreshed, the actions defined in the On Timer Refresh action 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.

10.7 Page Sources

The following actions are available in the Page Sources group of the Actions dialog (*screenshot below*):

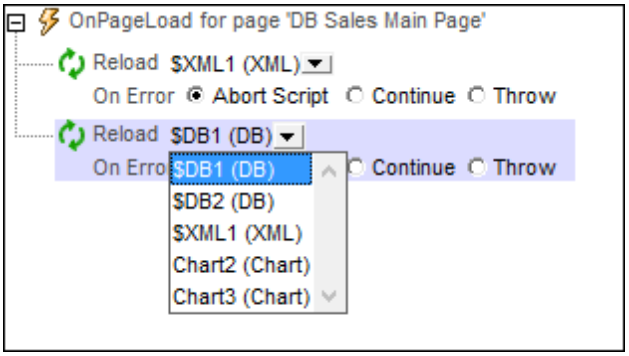
- Reload
- Load/Save File
- Load/Save Binary File
- Load/Save HTTP/FTP
- Load/Save String
- Load from SOAP
- Save
- Delete File/Folder
- Reset
- Execute SOAP Request
- Execute REST Request
- Get File Info
- Read Folder
- Save/Restore Page Sources



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 and Control Events*.

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:

- 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 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 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:

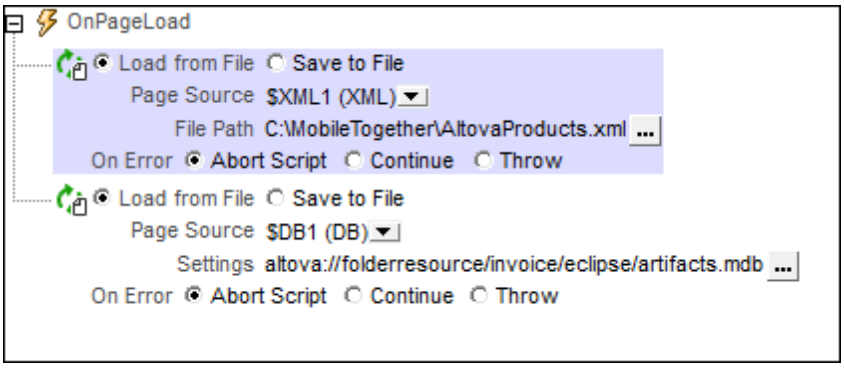
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select 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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

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*). You can then specify a file from which to load data for that page source when the event is triggered.



To load data for multiple page sources when the event is triggered, add multiple `LoadFromFile` actions, as shown in the screenshot above.

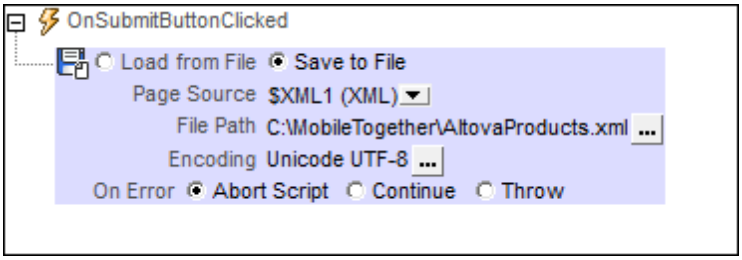
Error processing

The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

Save to file

Saves data from the page source that is selected in the action's combo box 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.



To save data from multiple data sources when the event is triggered, add multiple `SaveToFile` actions. To add another `SaveToFile` action, drag the `Load/SaveFile` action into the event tab, and set its radio button to the `SaveToFile` action.

Note: The `SaveToFile` action cannot be used to save to DBs. For saving to DBs, use the `Save` action and select one of the page source trees, or use the `DBExecute` action.

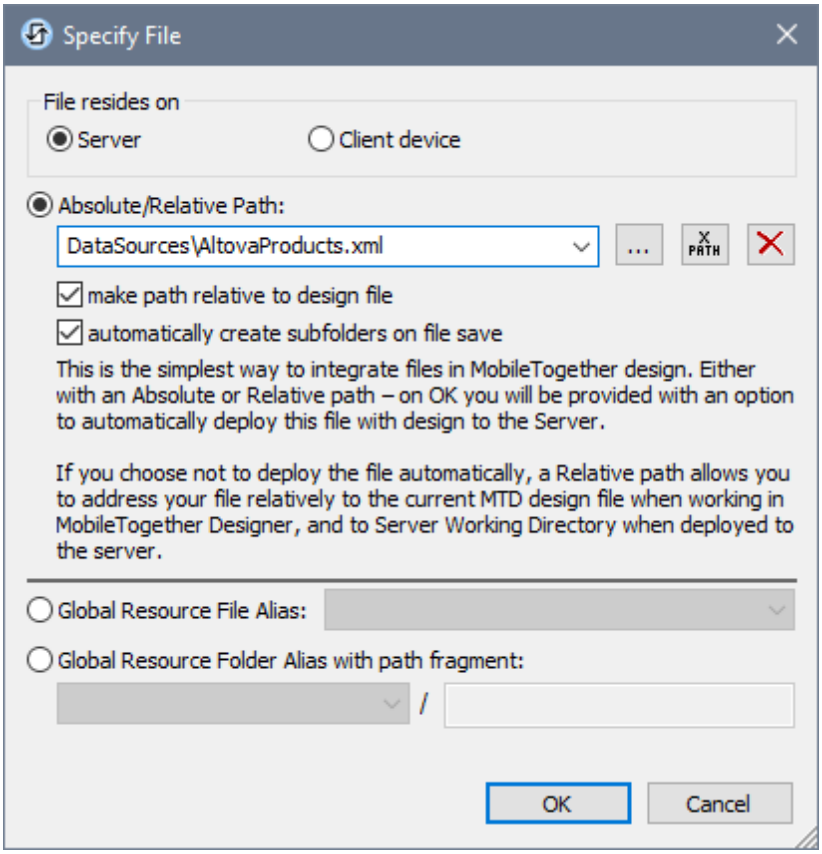
Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 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 Specify 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.

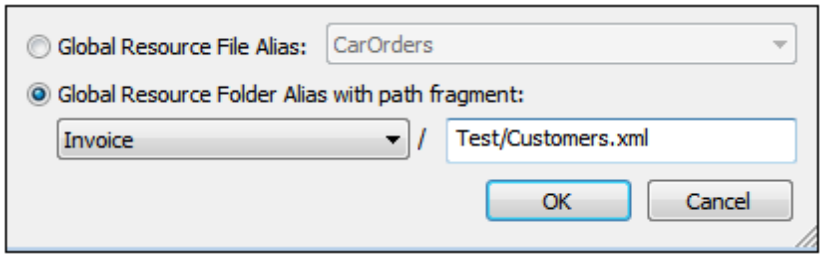


- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design 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 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 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 for details. 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 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.

- *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**). See the section Altova Global Resources for details.

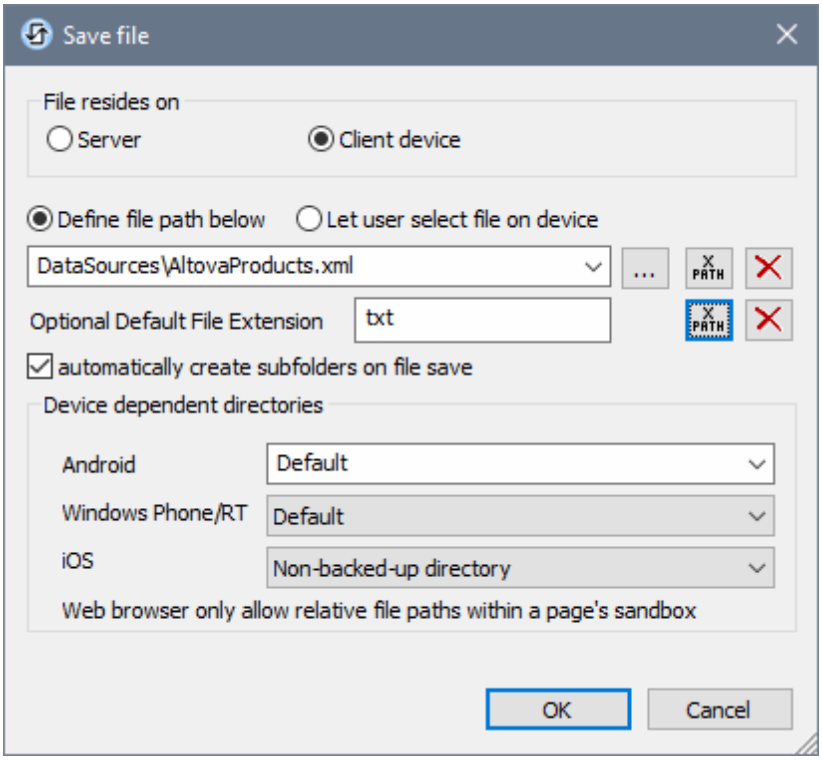
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



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**). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 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.



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 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 (*Source File* and *Target File* options), Load/Save File, Load/Save Image, and Load/Save Binary File.

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 (**Tools | Options**).

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:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

Load/Save Binary File

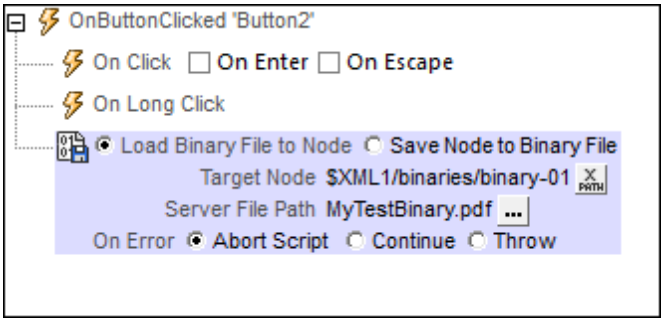
The Load/Save Binary File action enables the following:

- Loading a binary file into a data source node as Base64-encoded content
- Saving the Base64-encoded content of a data 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.

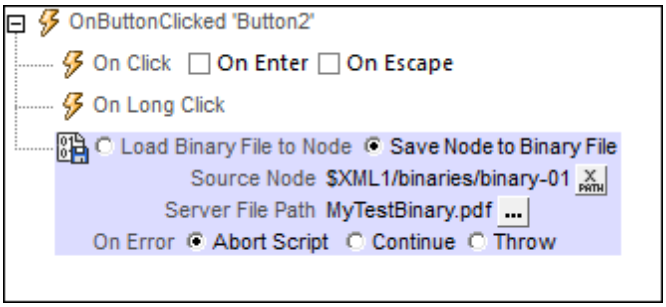
Loading a binary file into a data source node

A binary file can be loaded into a data 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 data 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.



Saving Base64-encoded content as a binary file

Base64-encoded content that is stored in a data 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 data 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).

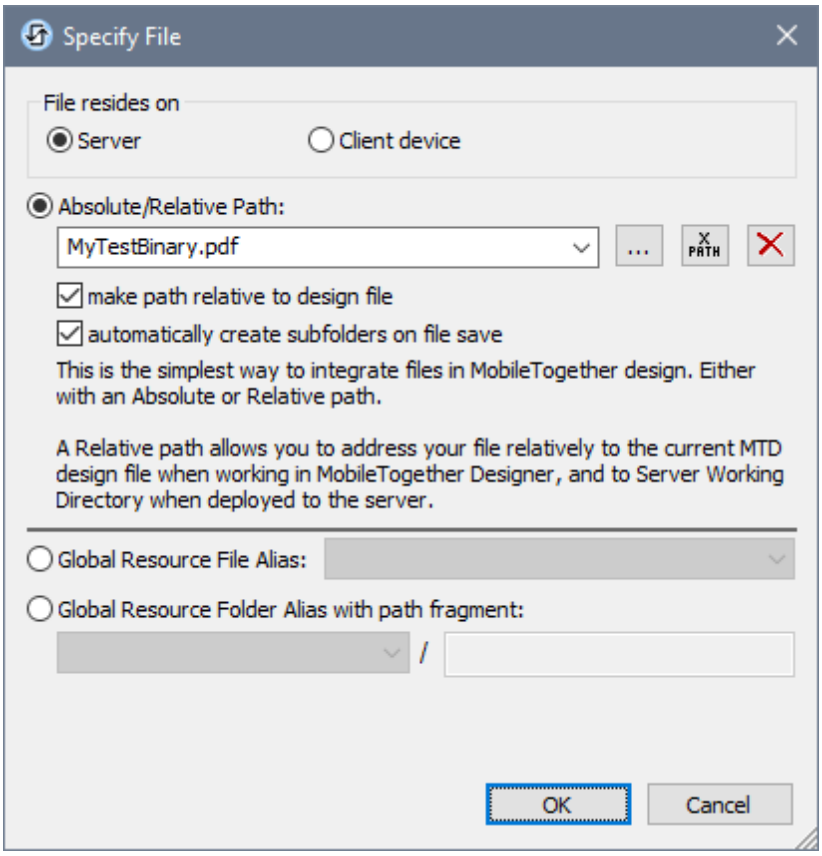


Binary file locations

When you click the **Additional Dialog** button of the *File Path* field of the Load/Save Image action (*see screenshots above*), the Specify 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 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.



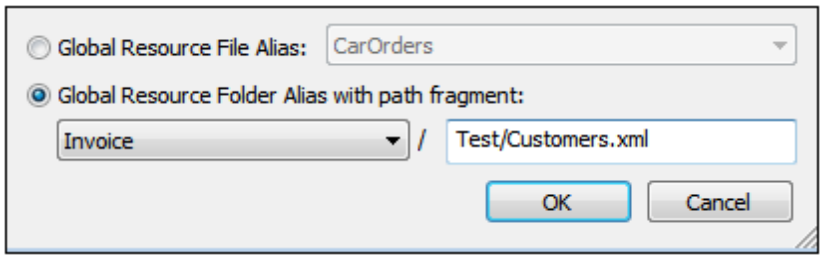
- *Absolute/Relative Path:* You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design 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 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 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 for details. 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 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.

- *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**). See the section Altova Global Resources for details.

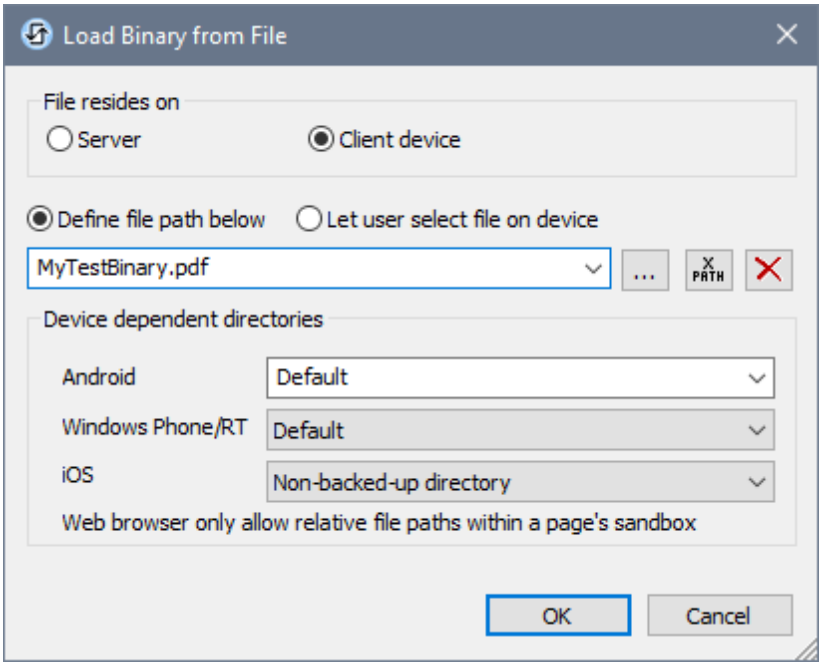
- *Global Resource Folder Alias with path fragment:* Select a folder alias from the folder aliases available in the combo box (*see screenshot below*).



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**). The path fragment specifies the rest of the path to the file resource. See the section Altova Global Resources 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.



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 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 (*Source File* and *Target File* options), Load/Save File, Load/Save Image, and Load/Save Binary File.

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 (**Tools | Options**).

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:

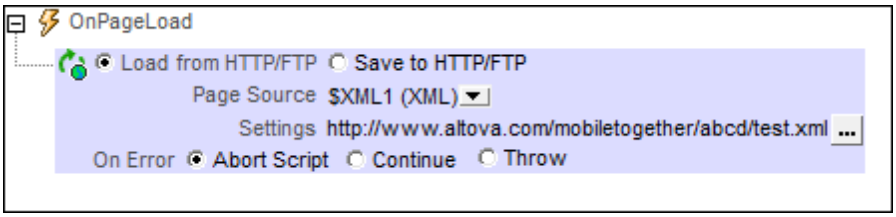
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select 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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

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. When the event is triggered, data from the HTTP/FTP source will be loaded into the page data source you have specified. To load data for multiple data sources, add multiple `LoadFromHTTP/FTP` actions.



Error processing

The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

Save to HTTP/FTP

Saves the data 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*). To enter access details of the HTTP/FTP location, click ... . This displays the Edit Web Access Settings dialog for selecting HTTP/FTP sources; here you can enter the file's URL and security settings.

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:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

Load/Save String

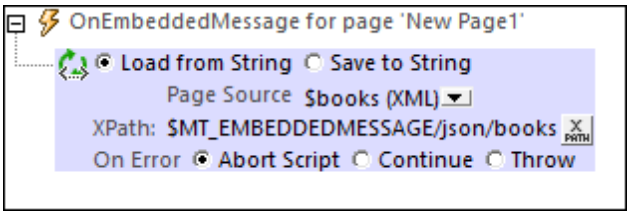
The two actions, **Load from String** and **Save to String**, respectively:

- 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, 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 action's XPath expression (*see screenshot below*), and generates the structure and data of the selected page source (*see screenshot*).

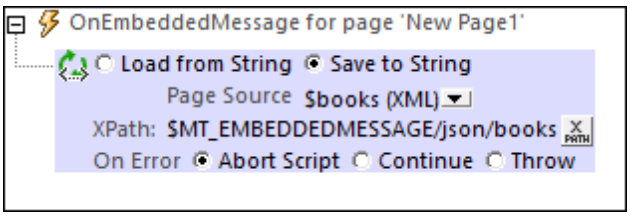


The type of the page source selected (JSON or XML) should correspond to the serialization of the string. So, if the targeted string is a JSON string, then a JSON document will be generated in the selected JSON page source (inside the page source's **json** element). When a serialized XML string is loaded into 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 serialized string:* In the screenshot above, an XML string is located in the JSON page source node **$MT_EMBEDDEDMESSAGE/json/books**. 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 need not take a string from a node; you can also enter the string directly in the XPath expression.)
- *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 has 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.

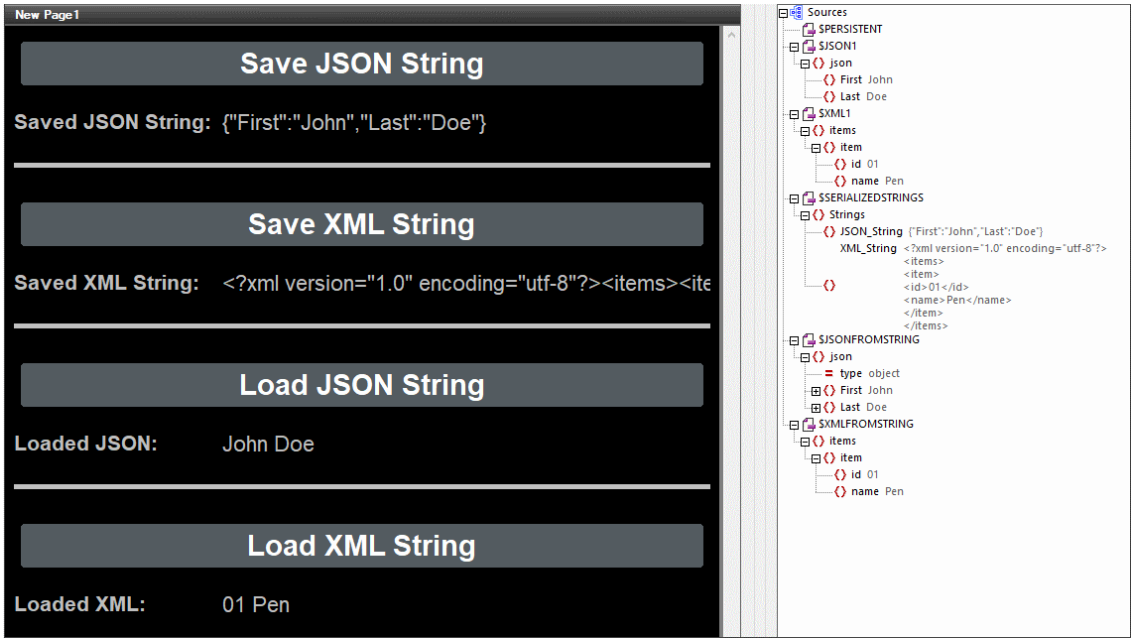


Note the following points:

- 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\MobileTogetherDesigner4\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 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 action to load the just-serialized strings into new page sources (**`$JSONFROMSTRING`** and **`$XMLFROMSTRING`**).

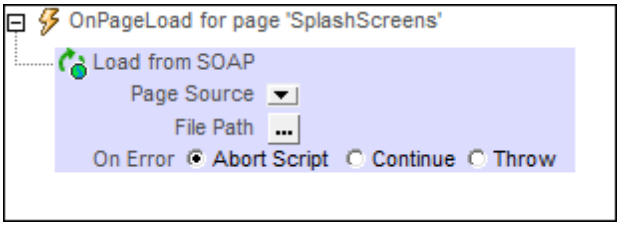


- The design file contains a JSON page source (**$JSON1**) and XML page source (**$XML1**), the structures and data of which are defined in the file itself (*see screenshot of simulation above*). Both data structures are rudimentary.
- The design contains a button to respectively save each page source as a serialized string (**Save JSON/XML String**), each in a different node of another page source named **$SERIALIZEDSTRINGS**. Both these buttons use the Save to String action.
- The generated serializations are used to create new pages sources, respectively, **$JSONFROMSTRING** and **$XMLFROMSTRING**. This is done by triggering the Load from String actions of the third and fourth buttons (**Load JSON/XML String**).

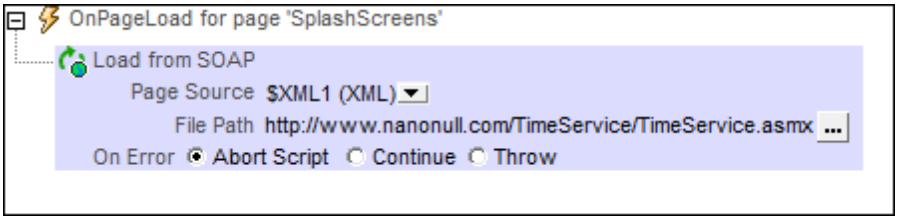
This example, in effect, uses the two actions (Save to String and Load from String) to make a round trip from one page source to another page source via a serialized string. The origin and destination page sources will have the same structure and data.

Load from SOAP

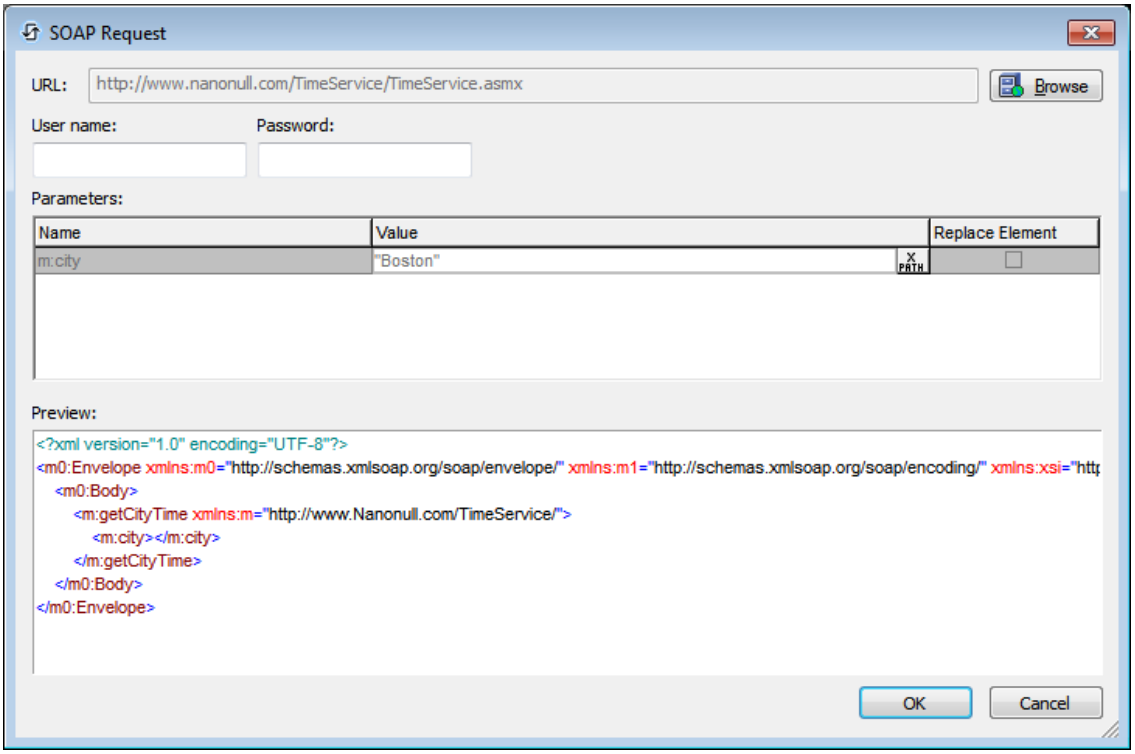
For each `LoadFromSOAP` action, you can load data from a SOAP request that is generated from a WSDL file. To choose the WSDL file, click the **Additional Settings** button of the *Source* field (*see screenshot below*).



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*).



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.



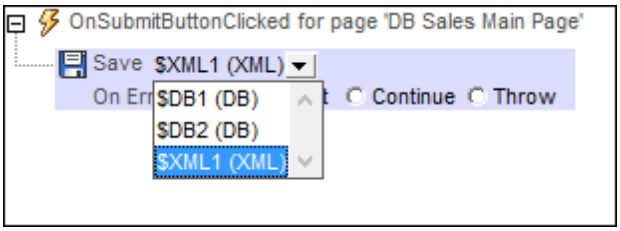
To load data from multiple data sources when the event is triggered, add multiple `LoadFromSOAP` actions.

Error processing
The *On Error* option lets you define what should be done if an error occurs:

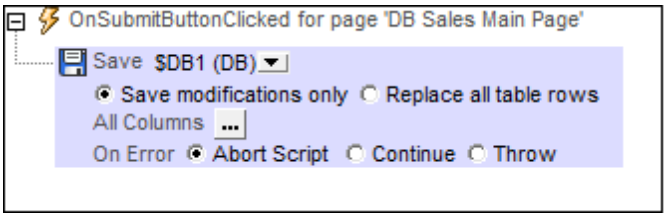
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select 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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

Save

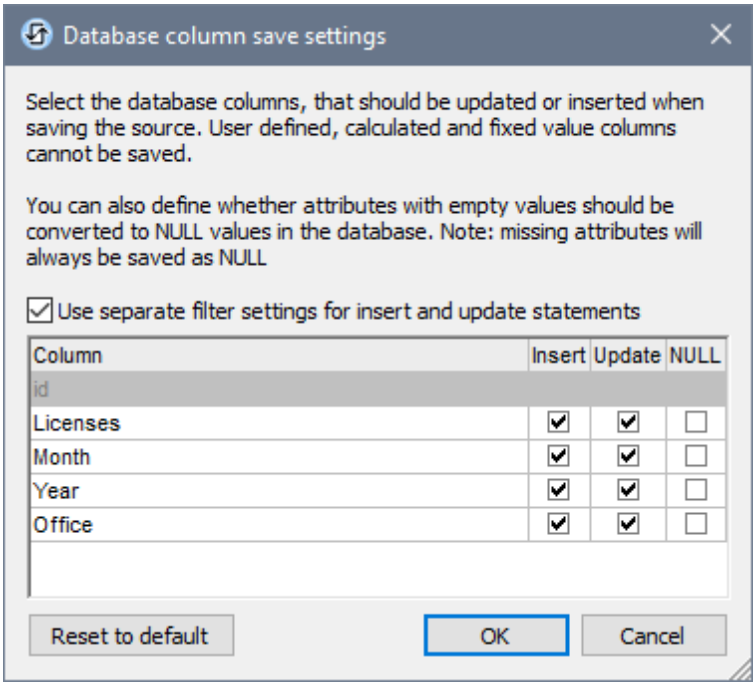
Saves data of the data source that is selected in the action's combo box to the default file of that data source. The data source must be an editable XML file or DB. To save data for multiple data sources, add multiple Save actions. Note that page sources that are read as JSON will also be saved as JSON (not as XML even though the data is presented in the GUI as an XML tree); *also see* *Page Source Options*.



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. 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.



To select specific columns to update, click ... . This displays the Database Column Save Settings dialog (*screenshot below*).



The dialog displays the columns of the DB data 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. If you wish to reset the *Save settings* so that all columns are updated, click **Reset to default**.

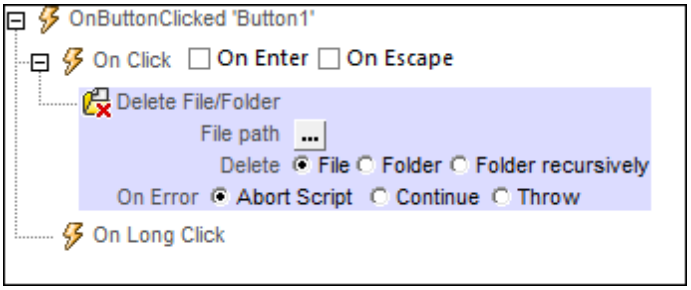
Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

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.



- *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.

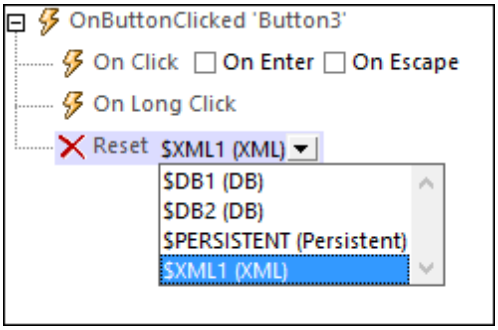
Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

For additional information about file locations on clients and server, see Tree Data and Load/Save Image.

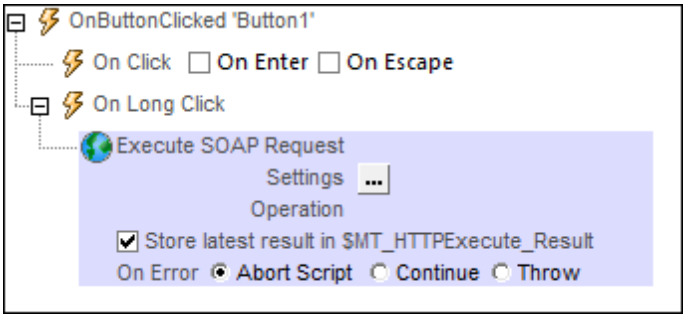
Reset

Resets the data source that is selected in the action's combo box to the default values defined in the Page Sources Pane. Note that you can reset any kind of data source that has been defined in the Page Sources Pane, including the **$PERSISTENT**, **$MT_GEOLOCATION**, **$MT_FILEINFO**, and **$MT_NFC** data source trees.

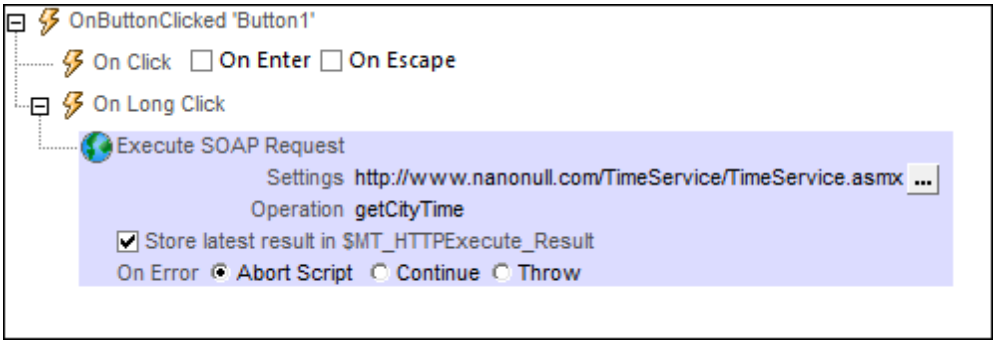


Execute SOAP Request

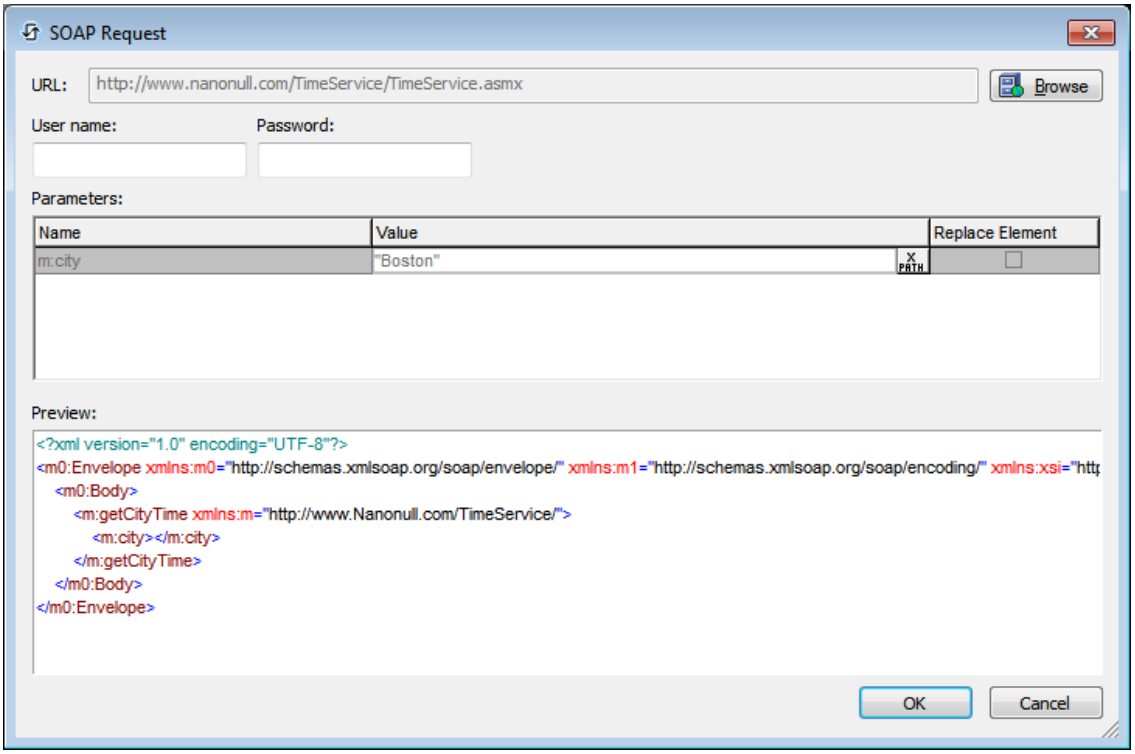
Executes a SOAP request that is generated from a WSDL file. To choose the WSDL file, click the **Additional Settings** button of the *Settings* field (*see screenshot below*).



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 `$MT_HTTPExecute_Result` variable. You can then use this variable to access the data in the SOAP response at some other location in the design. Note, however, that the `$MT_HTTPExecute_Result` variable can also be used by the Execute REST Request action. So the variable will contain the last result generated by *any* of the actions that use it.



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*).



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.

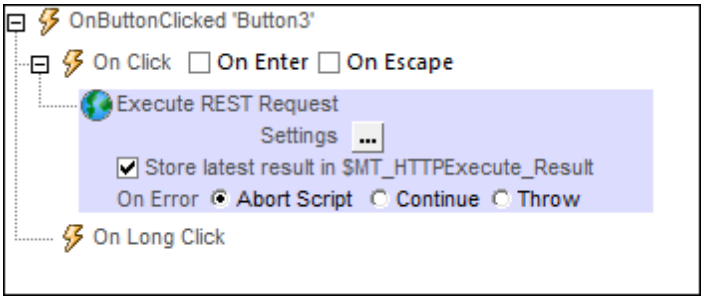
Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

The tutorial SOAP Requests shows how to use the Execute SOAP Request action.

Execute REST Request

Executes a REST request that you define in the RESTful API Request dialog. To define the REST request, click the **Additional Settings** button of the *Settings* field (*see screenshot below*).



After you have defined the REST request, the URL of the request is displayed in the *Setting*s field of the action. The request will be executed at runtime. If you wish to store the result of the request in the `$MT_HTTPExecute_Result` variable, check the *Store latest result* option (*see screenshot above*). You can then use the `$MT_HTTPExecute_Result` variable to access the result elsewhere in the design. Note, however, that this variable can also be used by the Execute SOAP Request action. So the variable will contain the last result generated by *any* of the actions that use it.

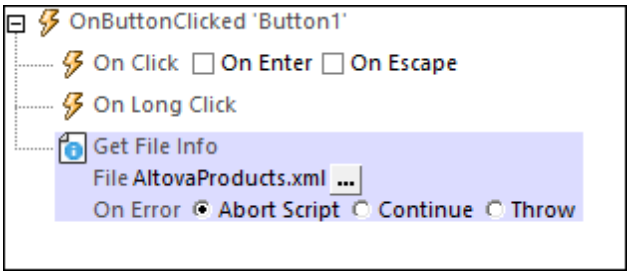
If you wish to change the REST request after one has already been defined, click the **Additional Dialog** button of the *Settings* field (*see screenshot above*). This displays the RESTful API Request dialog, in which you you can define the new request.

Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

Get File Info

The Get File Info action (*screenshot below*) adds information about the specified file (or folder) to a page source called **$MT_FILEINFO**. Note that there is only one **$MT_FILEINFO** page source per page. So, if the Get File Info action is triggered multiple times 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.

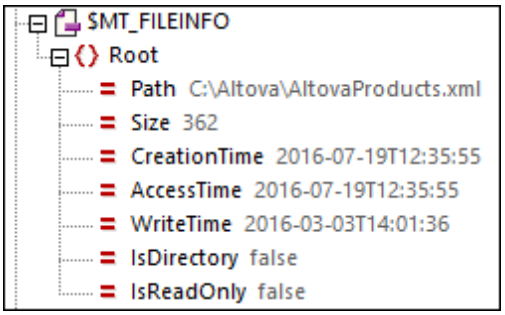


Note: The **$MT_FILEINFO**T page source is also created in the design

---

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 give below.

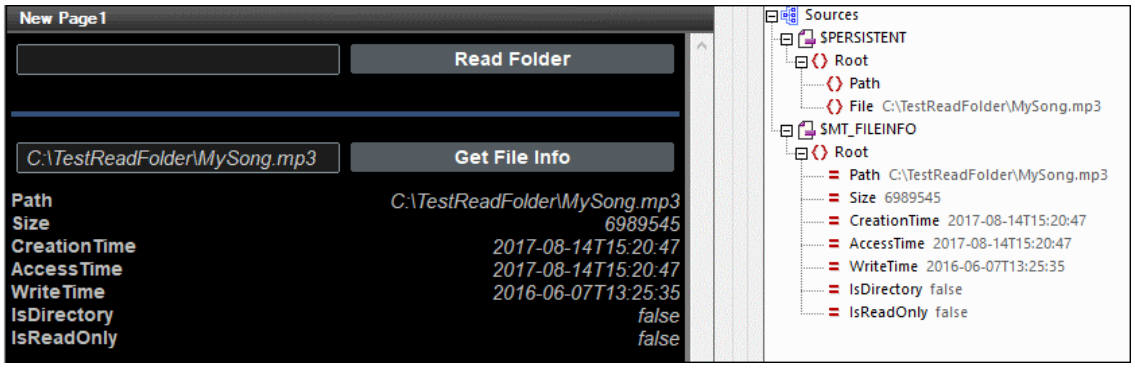


- *Path:* The full path of the file being reported.
- *Size:* The file size In bytes.
- *CreationTime:* The time when the file was created at its current location. If a file is copied to a new location, then the time at which it was copied will be the creation time. In such cases, the creation time could 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** page source, however, also has some other nodes in its structure: the repeating **File** children of **Root**. The **File** elements receive data obtained by another action—Read Folder—and are described there.

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\MobileTogetherDesigner4\MobileTogetherDesignerExamples\Tutorials\Actions`.

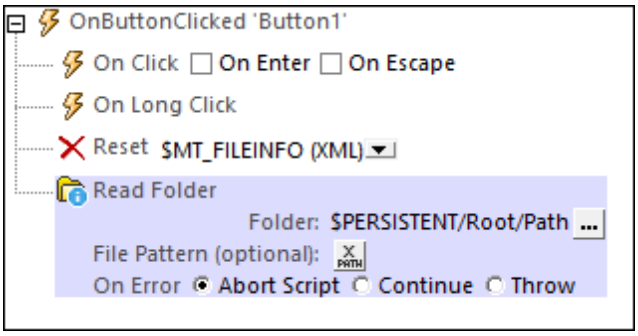


The example works as follows:

- 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.

Read Folder

When a Read Folder action (*see screenshot below*) is added to the design, the **$MT_FILEINFO** page source, which is structured as a set of repeating **File** elements, is created in the design. In the Read Folder action, you specify the folder that you want to read (*see screenshot and example below*). At run time, the contents of the specified folder (files and sub-folders) are read, and details of each item of the target folder are placed in a corresponding **File** element of the **$MT_FILEINFO** page source.

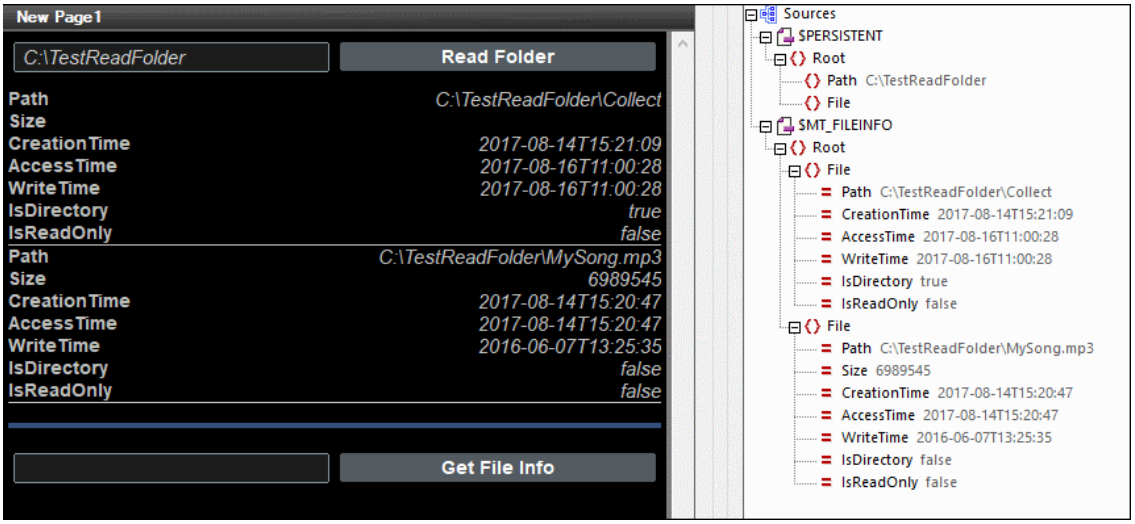


Note the following points:

- The folder can be specified in an XPath expression as a string or as an expression that locates a page source node. Only files in the specified folder are read. Files in sub-folders are **not** read.
- 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. For details of the folder itself, submit the name of the folder in the Get File Info action.
- Each page contains one **$MT_FILEINFO** page source. So, if there are multiple Read Folder actions on a page, then, at run time, the **$MT_FILEINFO** page source at any given time will contain information about the folder that was read by the last Read Action to have been triggered.
- The data read by the Read Folder action is passed to the **File** child elements of **$MT_FILEINFO/Root**. The attributes of **Root** receive information obtained by another action—the Get File Info action.
- Note that the **$MT_FILEINFO** page source is also created when a Get File Info action is added.

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\MobileTogetherDesigner4\MobileTogetherDesignerExamples\Tutorials\Actions`.

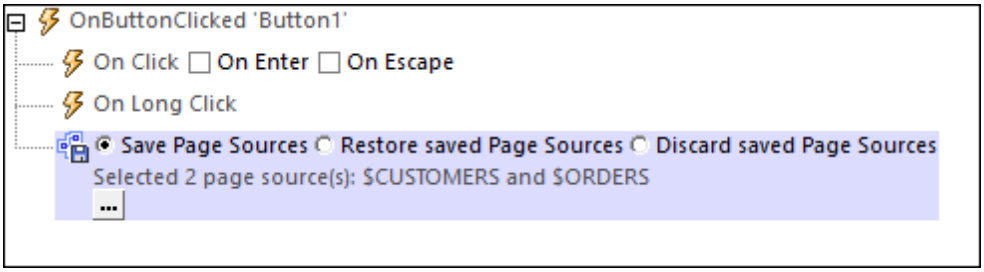


The example works as follows:

- 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.

Save/Restore Page Sources

Each Save/Restore Page Sources action consists of three sub-actions (*see screenshot below*), of which one may be selected.



The three sub-actions are:

- *Save Page Sources:* Save an internal copy of one or more page sources. Select the page source/s to save 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 Saved Page Sources:* After an internal copy of a page source has been saved (with *Save Page Sources*), a page source might be further modified. The *Restore Saved Page Sources* action returns the page source to the state of the most recent internally saved copy.
- *Discard Saved Page Sources:* After an internal copy of a page source has been saved (with *Save Page Sources*) and that page source has been further modified, the *Discard Saved Page Sources* action discards the previously saved internal copy.

Note: The Save/Restore Page Sources action applies only to page sources saved temporarily—that is, they are **not** saved to file. To save to file, use other actions like Save or Save File.

Usage

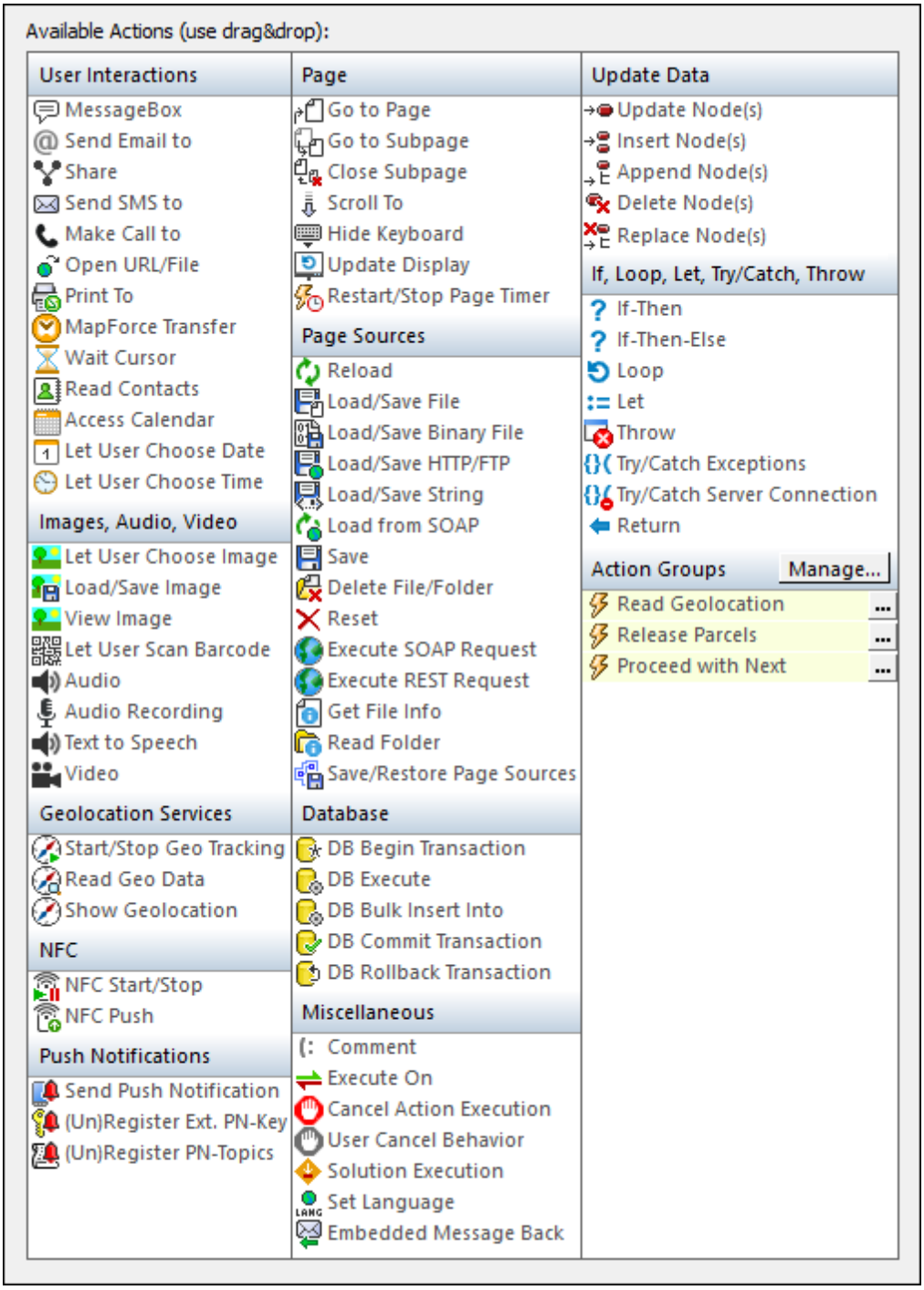
The Save/Restore Page Sources action enables you to save a page source temporarily, and then to accept or discard further modifications on the basis of whether one or more conditions are met. For example, before going to a subpage, a page source can be saved internally. After the user has edited data 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 saved page sources.

10.8 Database

The following actions are available in the Database group of the Actions dialog (*screenshot below*):

- DB Begin Transaction
- DB Execute
- DB Bulk Insert Into
- DB Commit Transaction
- DB Rollback Transaction

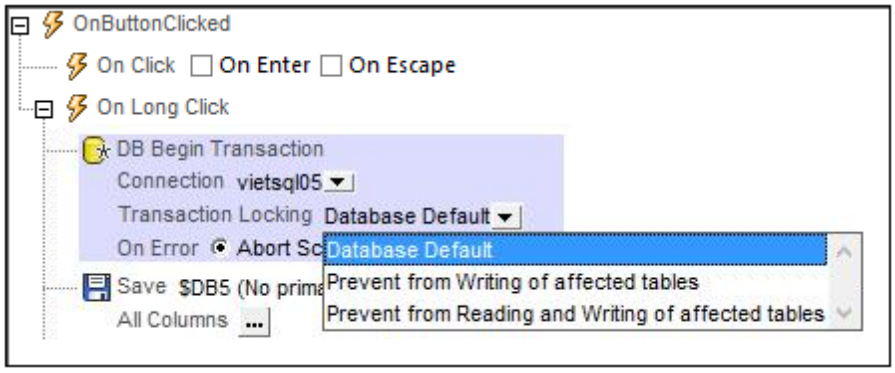
Note: These actions are used to *interact* with data in DB data sources. They are not suitable for displaying data. If you wish to display data from a DB data source, then insert (into the design) a control that is linked to a page data source. For information, see the sections about controls and data sources. The tutorials give you hands-on instructions about how to display page source data.



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 and Control Events*.

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.



The *Transaction Locking* option specifies the level 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.

Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 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.

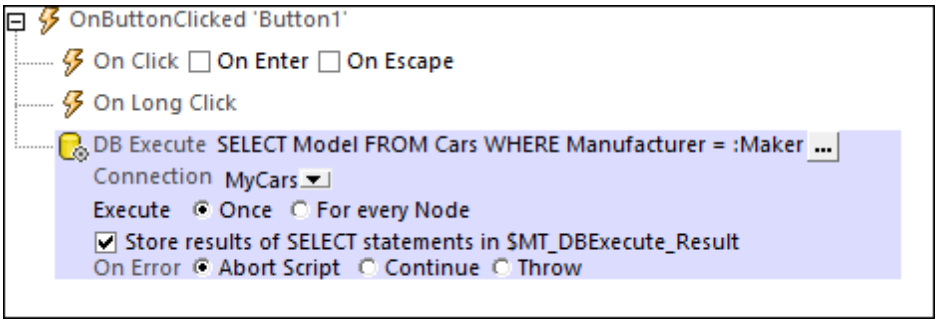
If you begin a transaction, all DB operations belonging to the same DB connection will use this transaction.

Committing a transaction makes changes visible to the world outside your transaction. Changes can be rolled back. 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.

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` variable. This variable can then be used in XPath expressions elsewhere on the page to provide the result of the DB Execute action.



Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

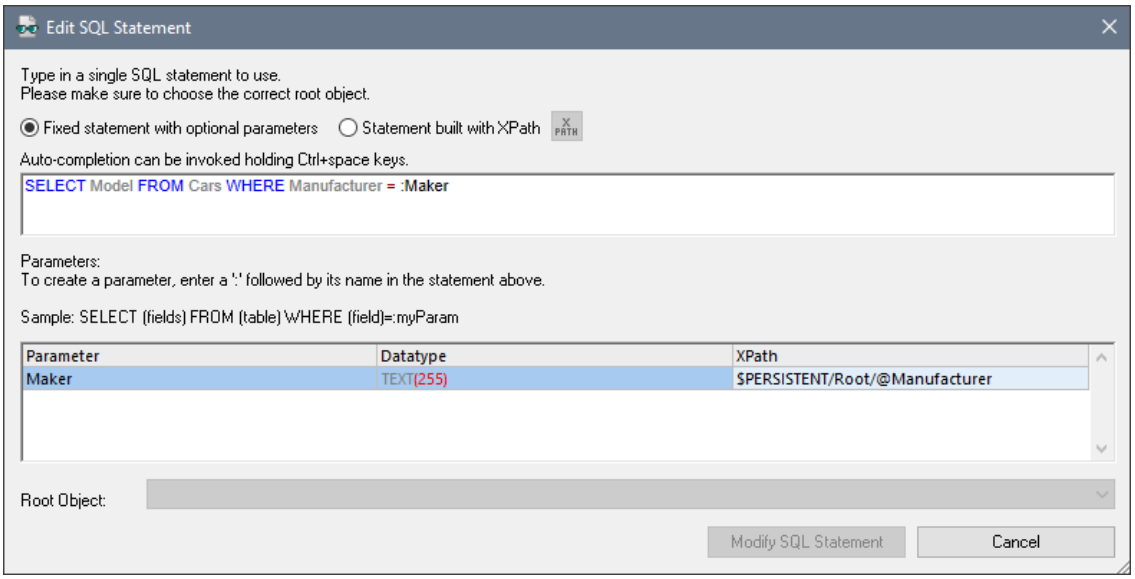
For more information about using the action, see the section Page Design | Database | The DB Execute Action.

Note: The DB Execute action is used to *interact* with data in DB data sources. It is not the ideal mechanism for displaying data. If you wish to display data from a DB data source, then insert (into the design) a control that is linked to a page data source. For information, see the sections about controls and data sources. The tutorials 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.



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 data 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.

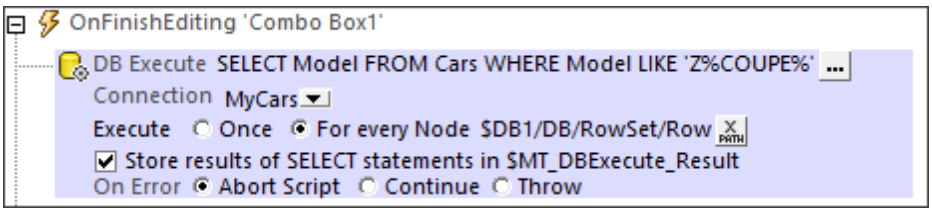
Statement built with XPath

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 that appears, enter the XPath expression and click **OK**.

Execute once or for every node

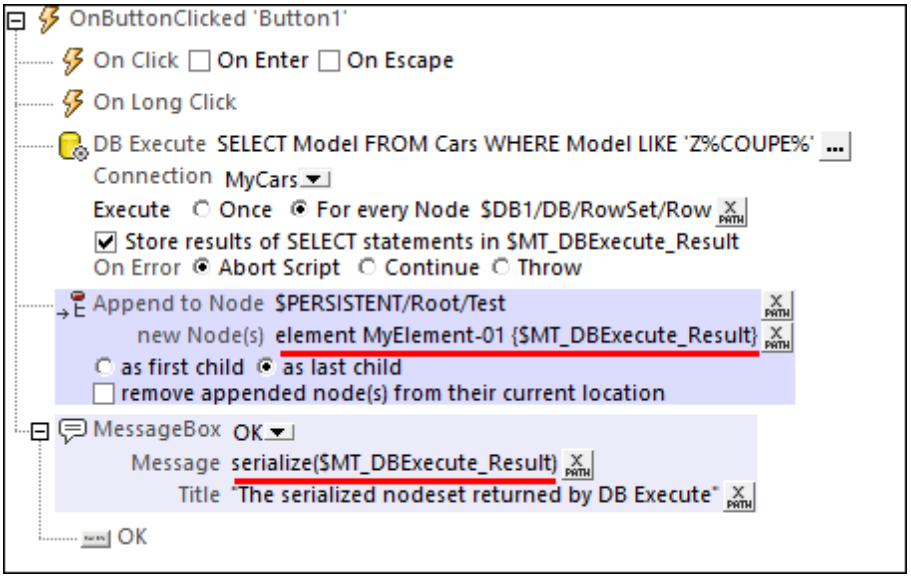
The SQL statement can be executed once on the data source, or it can be executed on all the nodes of a custom defined nodeset. If you select the latter option, you need to enter an XPath expression that generates the nodeset. The SQL statement will then be executed for each node in this nodeset. Additionally, you can query the value of the current node of the nodeset by using the variable `$MT_TargetNode`. This variable can be used, for example, in the definition of a parameter used in the SQL statement (*see "SQL statements with parameters" above*).



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.



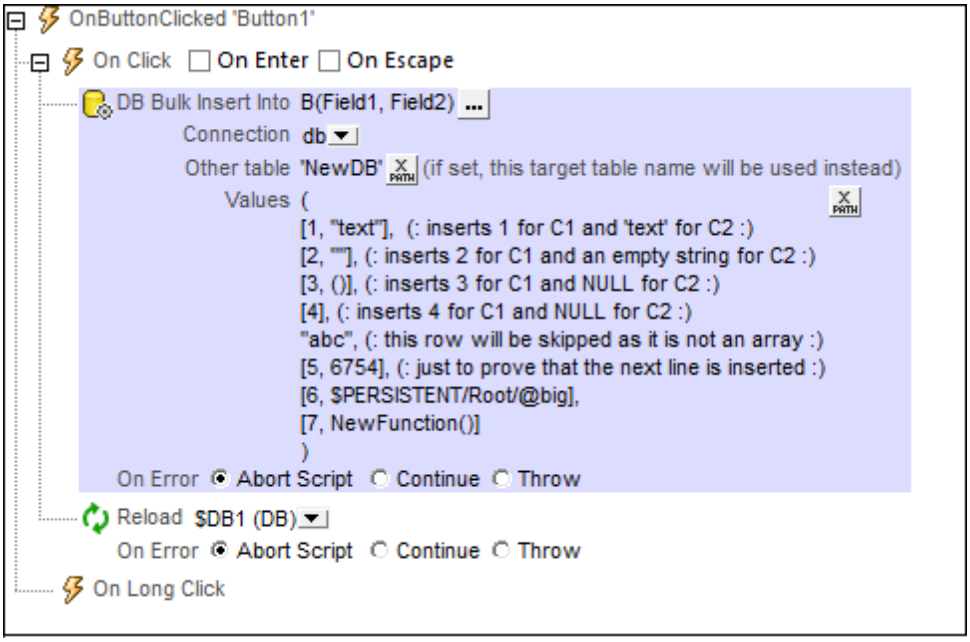
If you want to access a specific node in the nodeset, you can access it with an XPath expression. For example, consider the DB Execute action in the screenshot above. Let us say that the **SELECT** statement returns the following nodeset:

```
<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')]]`.

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*).



- *DB Bulk Insert Into*: When selecting the DB table into which to insert, you specify the DB connection method 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.

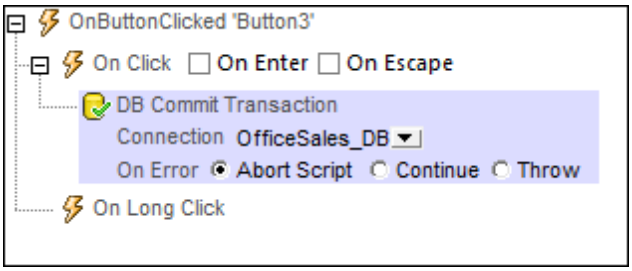
In the screenshot above, we have used a Reload action to update the DB page source that contains the modified table.

Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

DB Commit Transaction

When the event is triggered the DB Commit Transaction action commits a transaction to 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.



Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 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, all DB operations belonging to the same DB connection will use this transaction.

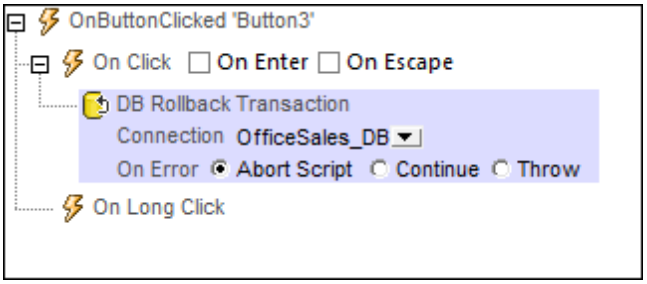
Committing a transaction makes changes visible to the world outside your transaction. Changes can be rolled back. In this case, even if you have done a Save on your page source, the changes won't be visible after a rollback! Note that any transaction that is not closed (committed or rolled back) when 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.

DB Rollback Transaction

When the event is triggered the DB Rollback Transaction action rolls back a transaction 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.



Error processing
The *On Error* option lets you define what should be done if an error occurs:

- *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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an additional option: *On Cancel*, which will be executed if the user cancels the dialog.
- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 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, all DB operations belonging to the same DB connection will use this transaction.

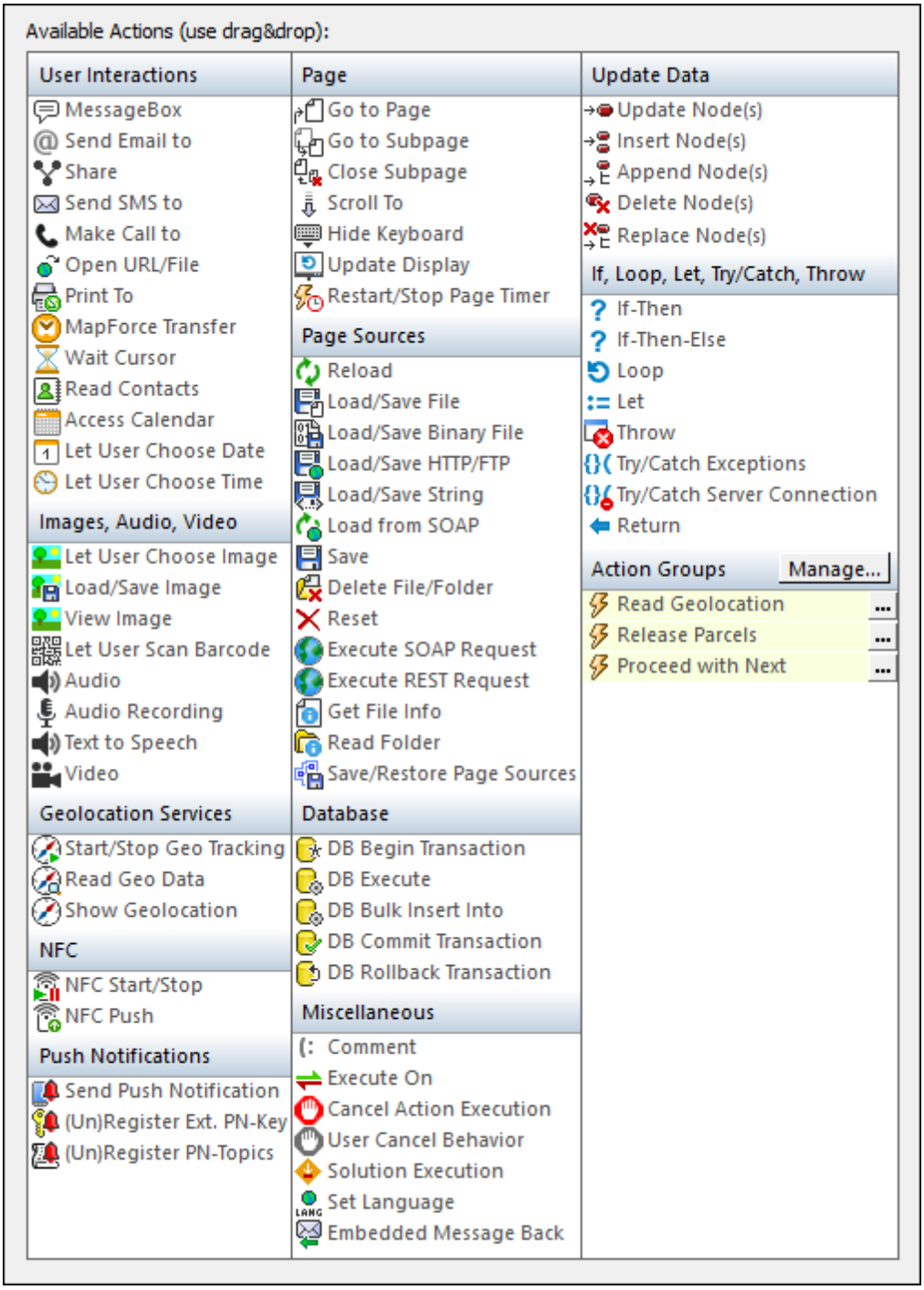
Committing a transaction makes changes visible to the world outside your transaction. Changes can be rolled back. In this case, even if you have done a Save on your page source, the changes won't be visible after a rollback! Note that any transaction that is not closed (committed or rolled back) when 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.

10.9 Miscellaneous

The following action is available in the Miscellaneous group of the Actions dialog (*screenshot below*):

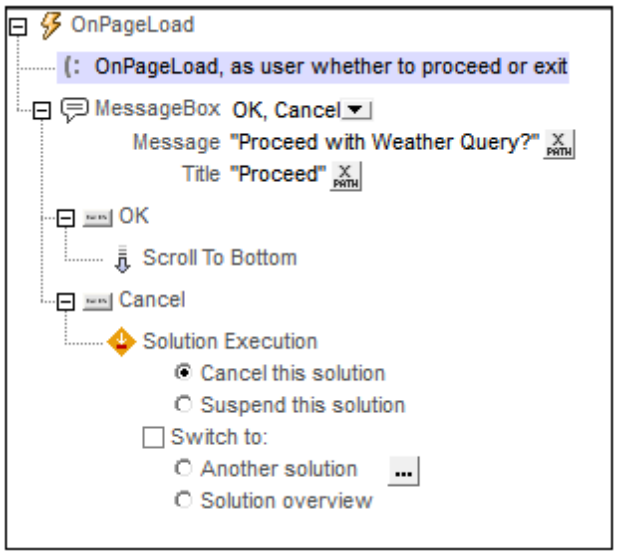
- Comment
- Execute On
- Cancel Action Execution
- User Cancel Behavior
- Solution Execution
- Read Contacts
- Set Language
- Embedded Message Back



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 and Control Events*.

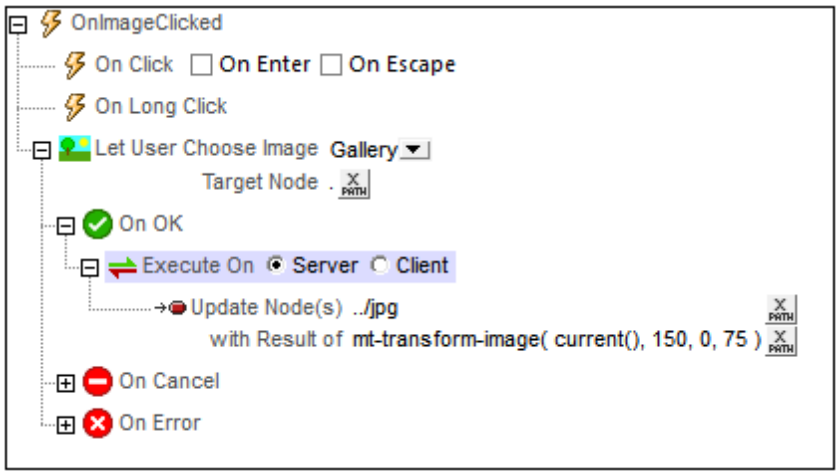
Comment

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.



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.



The screenshot above displays how the Execute On action is typically used:

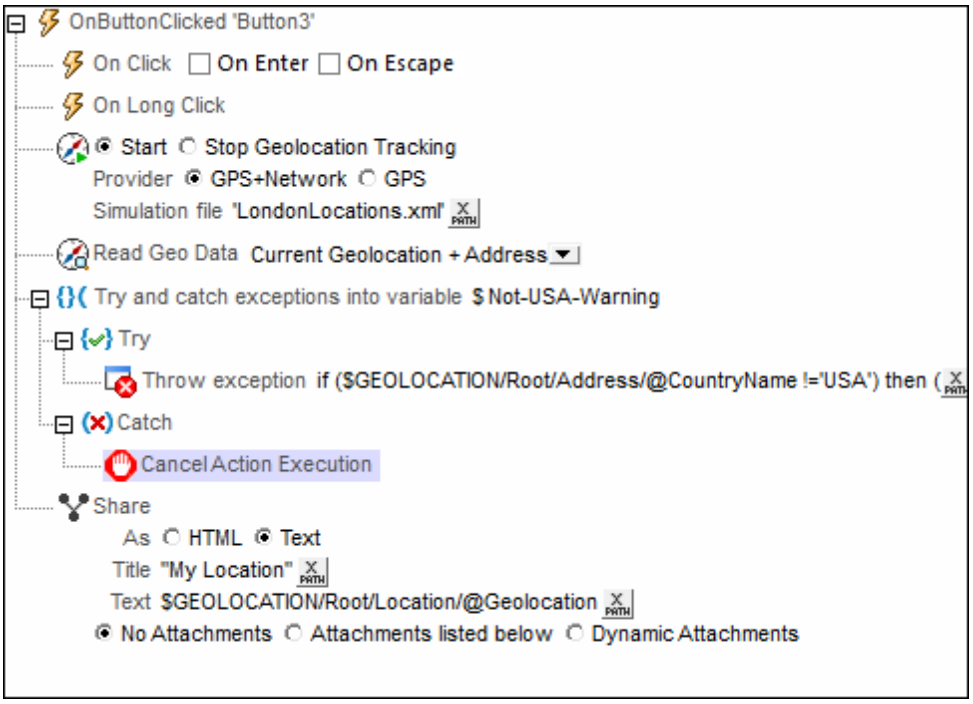
1. A Let User Choose Image action prompts the user to select an image from a gallery and save the image as Base64 to the current node (which is located, say, with `//image/base64`).
2. If the image is successfully transferred to the current node, the *On OK* condition uses the Execute On action to transform the user-selected image on the server (with the Altova XPath extension function **`mt-transform-image`**) 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`** 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, specifying that the child actions be performed on the server. All the child actions of this Execute On action will then be carried out on the server. You can use an action such as Update Node 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.

Cancel Action Execution

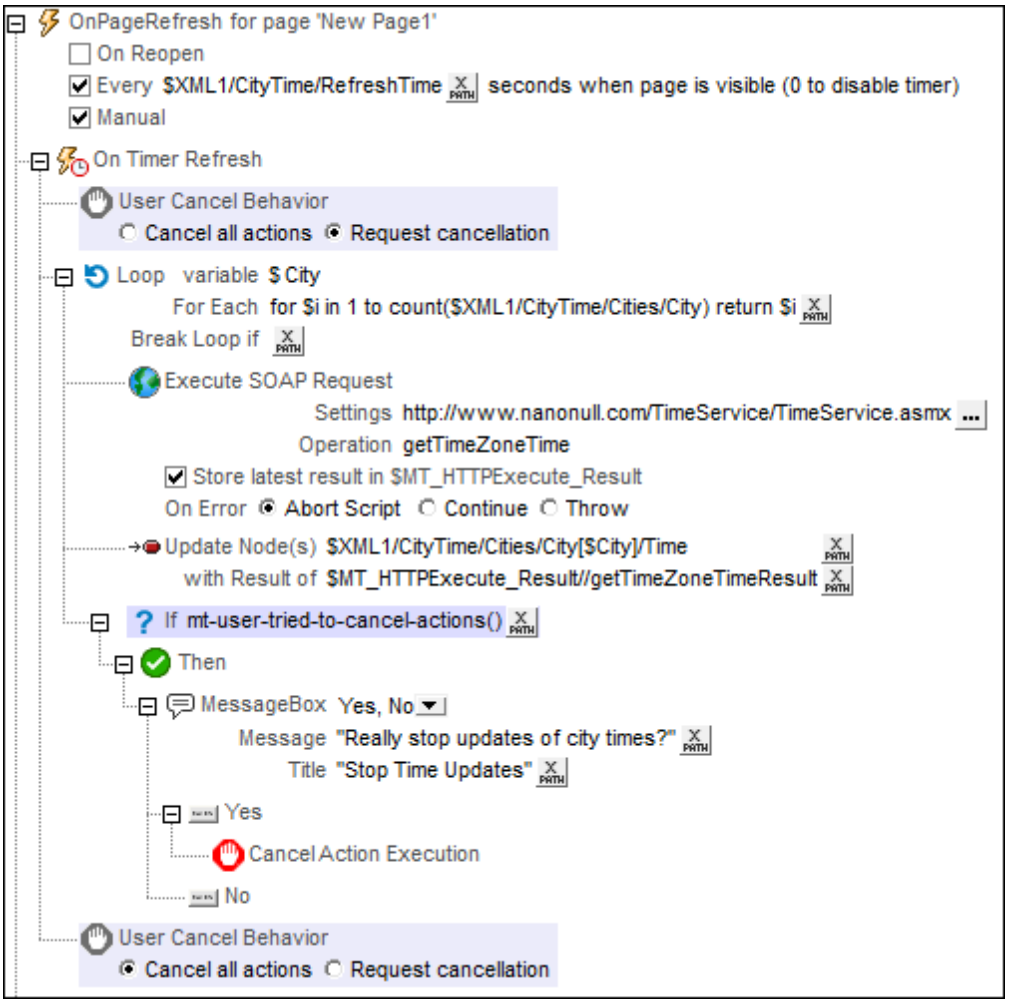
The Cancel Action Execution (*highlighted in the screenshot below*) cancels the execution of the event's action sequence. In the screenshot below, for example, if an exception is thrown in Try part of the Try/Catch action, then the Catch part of the Try/Catch action is executed. Since this part contains the Cancel Action Execution action, the event's sequence of actions is canceled. As a result, the Share action will not be executed.



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`** 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`** function to its default value of **`false`**.

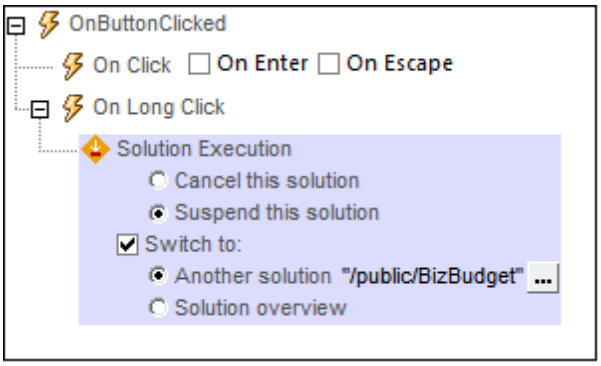


In the example shown in the screenshot above, the following is defined for the *On Timer Refresh* event option:

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`** function will be set to **`true`**.
2. A Loop action 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`** 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.

Solution Execution

When the event is triggered, the Solution Execution action provides you with the option to either cancel the solution or leave it suspended (that is, running in the background). In both cases (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 a 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.
Solutions 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.
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; only the active solution is supported.
- 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` event are executed, and audio playback that was paused is resumed. *Also see the project property On Switch to Other Solution.*
- The solution to switch to is specified by clicking the Edit XPath Expression button and

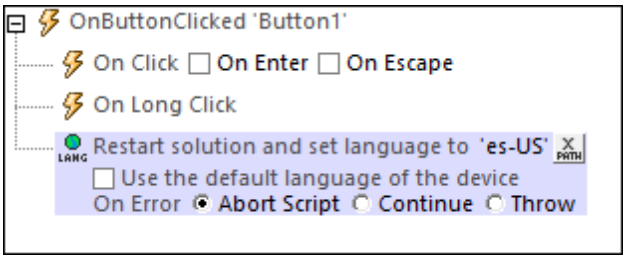
entering the location of the solution as a string (that is, within quotes). The location must be on the same server as the current solution and the location string must be exactly the same as the string that was entered when the target solution was deployed. *See screenshot above*. If the target solution is already running (minimized), then it is opened and continues from where it was when it was switched to its minimized state.

- 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.

Set Language

If a solution is localized, 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 (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 language setting of the mobile device must correspond to the code name of one of the solution's localization languages.

The Set Language action enables the solution to be restarted with a user-selected language—without needing to change the language of the device. For example, the user can tap a button to change to a specific language (*see screenshot below, which causes the solution to restart in US Spanish*), or the user can select a language from a dropdown list in a combo box.



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, 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** command, and then run a simulation.

Error processing
The *On Error* option lets you define what should be done if an error occurs:

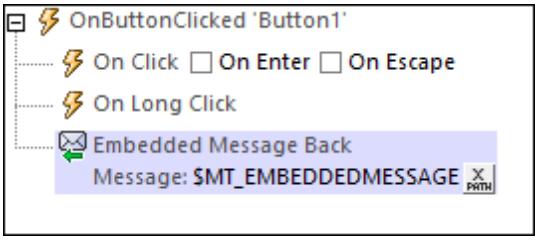
- *Abort Script:* After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- *Continue:* Actions are not terminated. Instead, you can select 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. For the actions Load/Save File, Load/Save Binary File, Load/Save Image, there is an

additional option: *On Cancel*, which will be executed if the user cancels the dialog.

- *Throw:* If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action 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 for details.

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.



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:

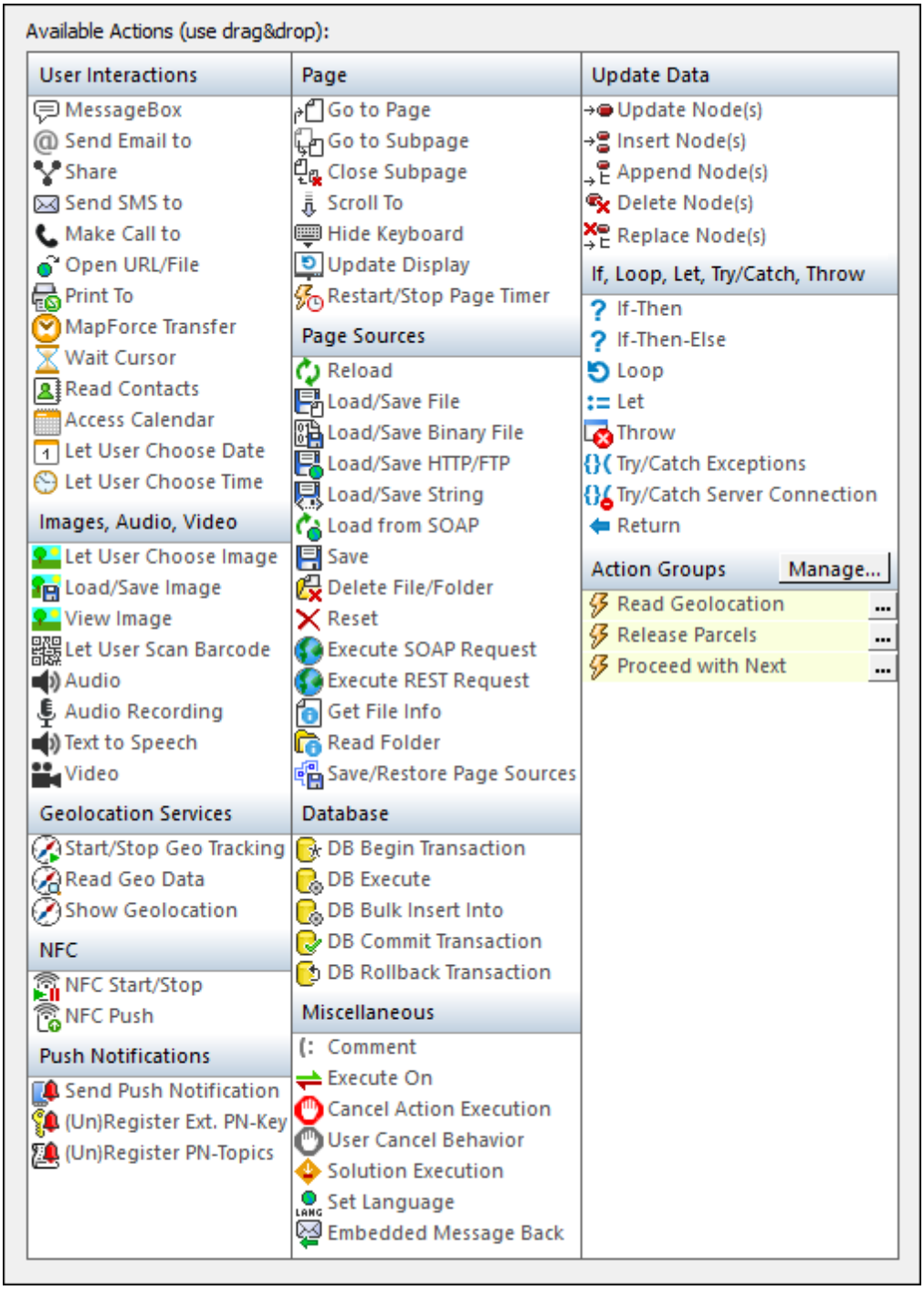
- **$MT_EMBEDDEDMESSAGE**, which is the page source tree that contains the JSON data to be processed and transmitted. Note that the root element of this tree is always named **json**. If the entire **$MT_EMBEDDEDMESSAGE** tree is returned, as shown in the screenshot above, then the serialized JSON string 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

10.10 Update Data

The following actions are available in the Update Data group of the Actions dialog (*screenshot below*):

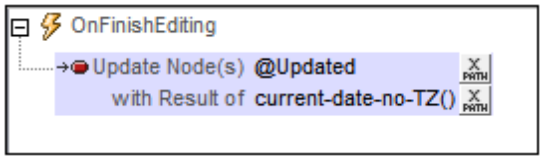
- Update Node(s)
- Insert Node(s)
- Append Node(s)
- Delete Node(s)
- Replace Node(s)



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 and Control Events*.

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()`).



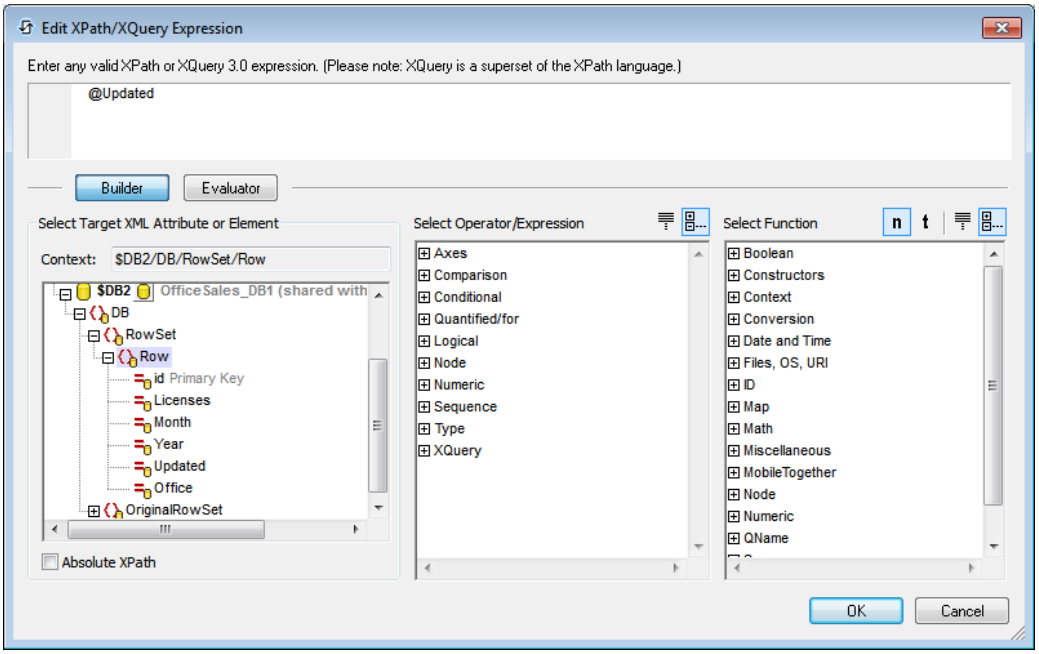
There are a few points to note when specifying the update node/s and update value:

- Importance of context node for relative paths
- The target node for the update can be referenced with the **`$MT_TargetNode`** variable
- 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.

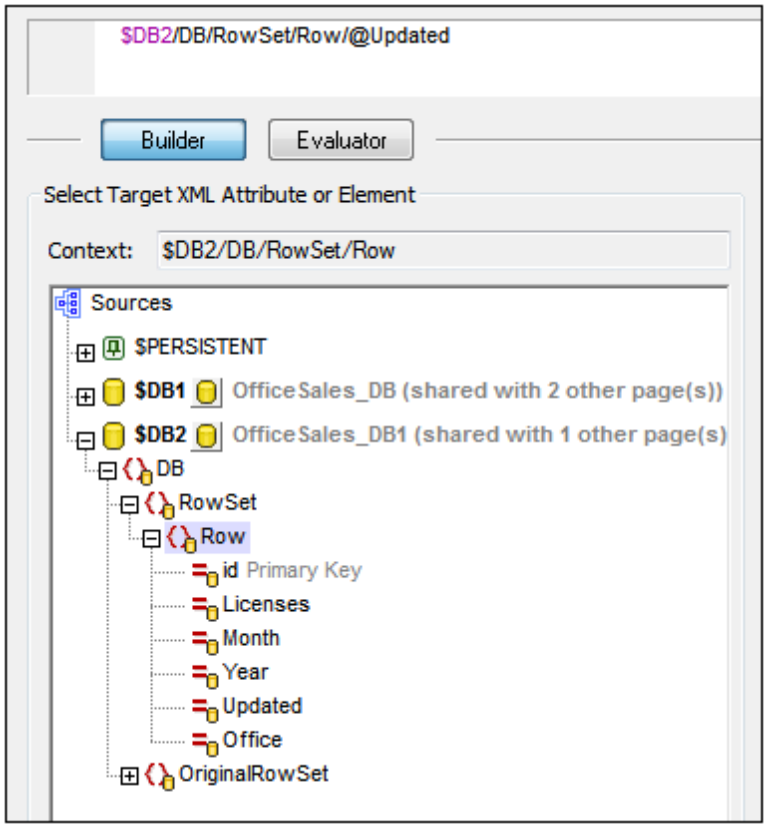
Importance of context node for relative paths

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.



- 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.



- Notice from the screenshot above, that you have access via XPath expressions to all the

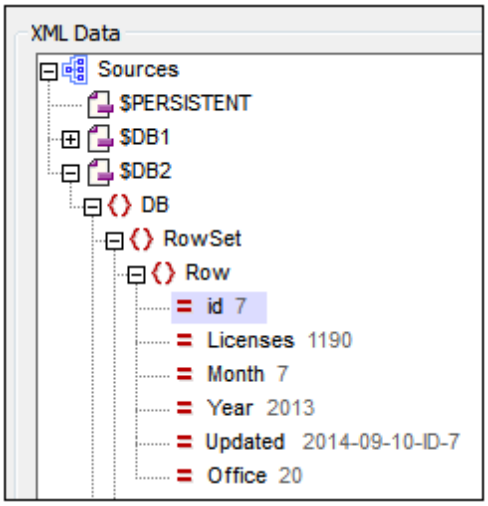
nodes of all data sources.

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.



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:

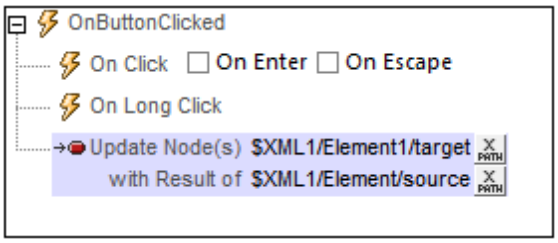
```
Update Node : $DB2/DB/RowSet/Row/@Updated
Update Value: concat(current-date-no-TZ(), '-ID-', $MT_TargetNode/../@id)
```

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.

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. Consider the Update Node(s) action defined in the screenshot below.



If the XML tree had the following structure and content:

```
<Element1>
   <source>AAA
      <subsource>BBB</subsource>
  </source>
  <target></target>
</Element1>
```

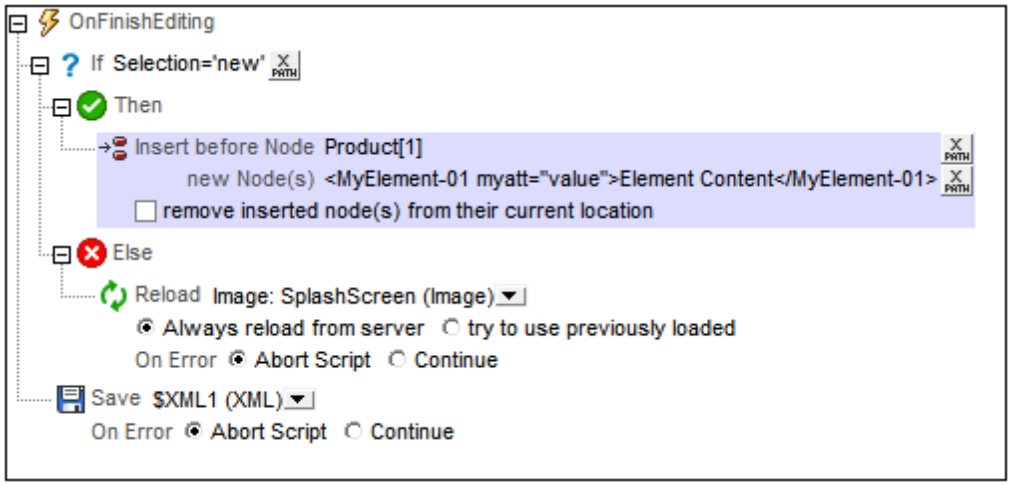
Then the `target` element would be updated with the text content of the mixed-content element `source`, while ignoring the content of its child element `subsource`. The node named `target` will be updated to `<target>AAA</target>`.

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.

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.



Note: The difference between Insert Node(s) and Append Node(s) is that Insert Node(s) adds the node/s before the selected node/s, whereas Append Node(s) adds the node/s as (first or last) child nodes of the selected node/s.

Location of inserted node/s

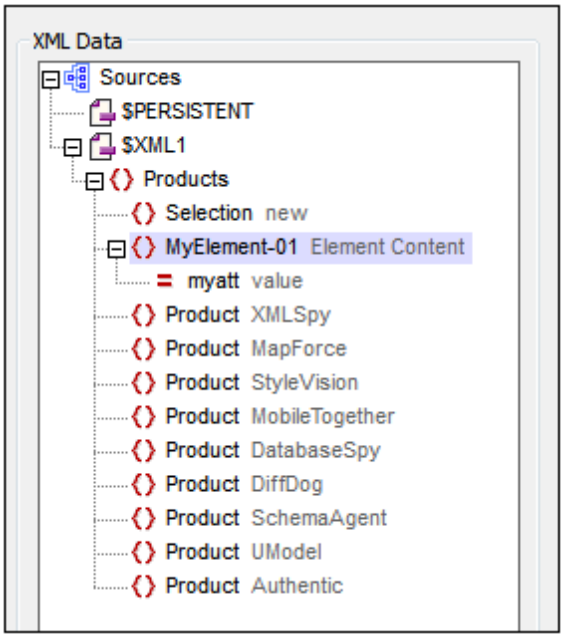
The new node/s are inserted before the node/s returned by the XPath expression for this setting (*Insert before Node*). In the screenshot above, the new node/s are inserted before the first `Product` element (selected with the XPath expression `Product[1]`). The context node is the parent of `Product`, 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.



You can also use an XPath locator expression to insert a node (and all its descendants) from a data source on the page. For example:

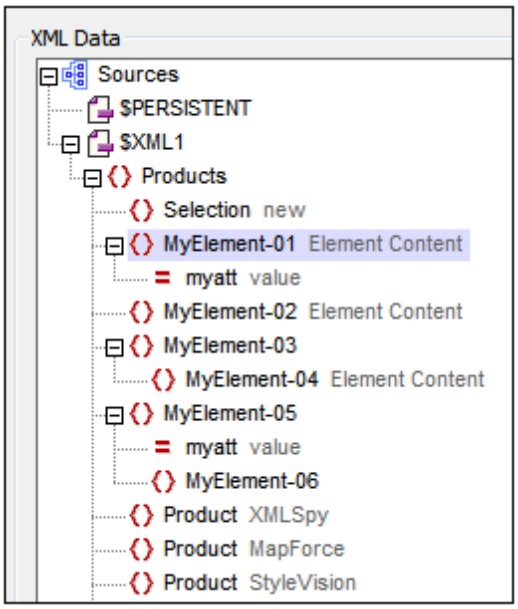
```
$XML2/Row
```

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{}}
```



Removing inserted nodes from their original locations

If the inserted node/s are obtained from one of the project's data 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 data sources—then selecting this option will have no effect on the data sources.

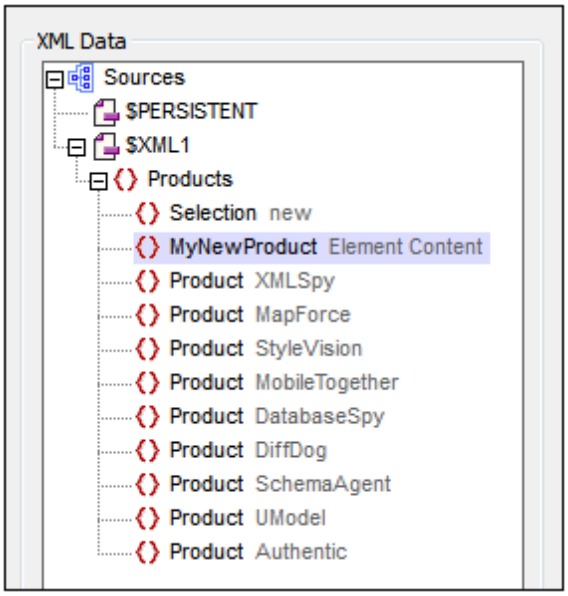
The $MT_TargetNode variable

The node in the Insert Node(s) definition that is targeted as the node before which to insert 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.

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.

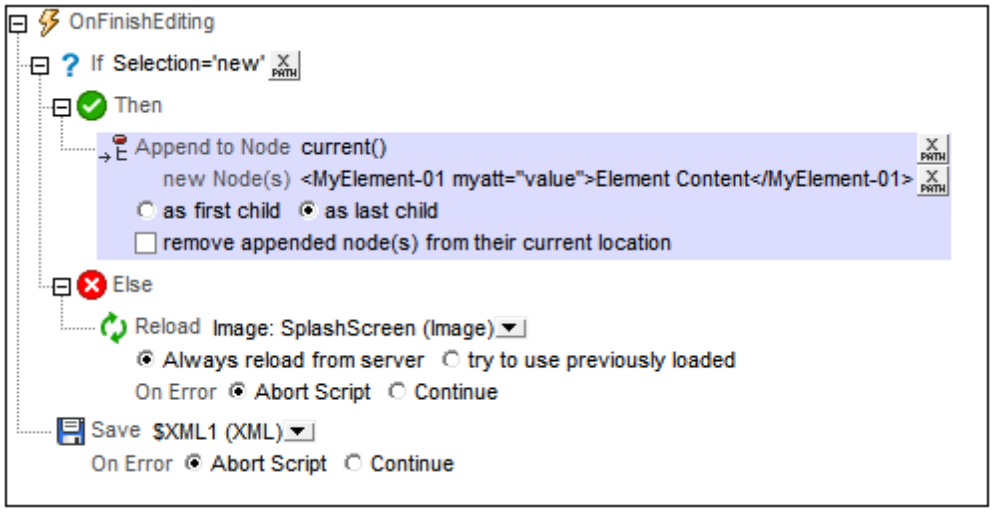
```
element {concat("MyNew", name($MT_TargetNode))} {"Element Content"}
```

The result of the Insert Node(s) action defined above is shown in the screenshot below.



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.



Note: The difference between Insert Node(s) and Append Node(s) is that Insert Node(s) adds the node/s before the selected node/s, whereas Append Node(s) adds the node/s as (first or last) child nodes of the selected node/s.

Location of appended node/s

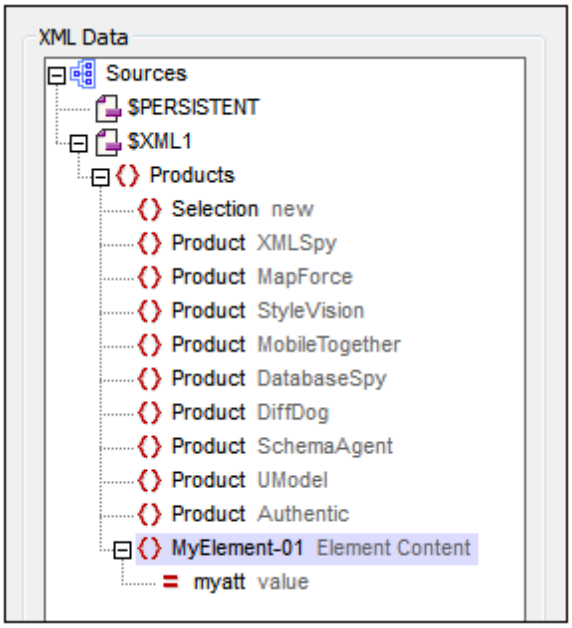
The new node/s are appended as the first or last child node/s of the node/s returned by the XPath expression for this setting (*Append to Node*). In the screenshot above, the new node/s are appended as the last child nodes of the context node, the `Products` element (selected with the XPath expression `current()`). To select whether the new node/s should be appended as the first or last child nodes/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 data source on the page. For example:

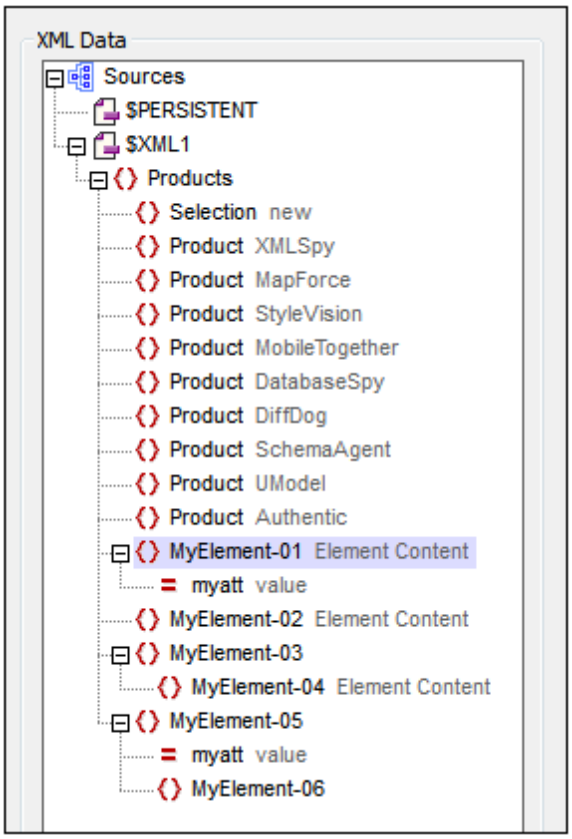
```
$XML2/Row
```

XQuery's computed node constructors can also be used, for example:

```
element MyElement-01 {xs:string("Element Content")}
attribute myatt{"value"}
```

The 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).

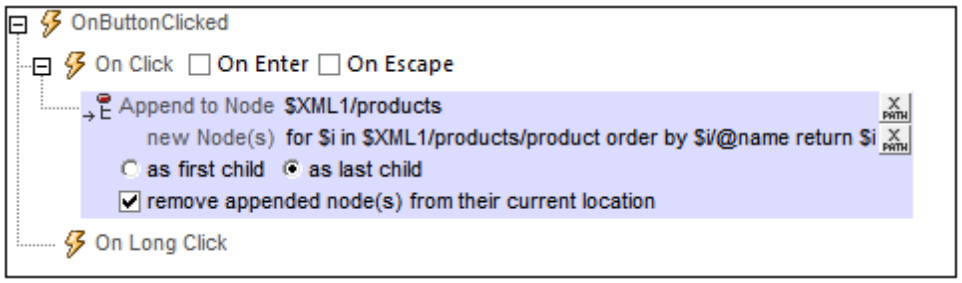
```
<MyElement-01 myatt="value">Element Content</MyElement-01>,
element MyElement-02 {"Element Content"},
element MyElement-03 {element MyElement-04 {"Element Content"}},
element MyElement-05{attribute myatt{"value"}, element MyElement-06{}}
```



Removing appended nodes from their original locations

If the appended node/s are obtained from one of the project's data 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 data sources—then selecting this option will have no effect on the data sources.

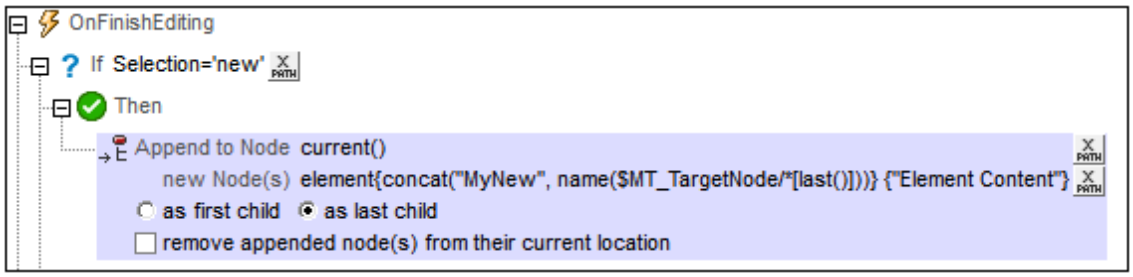
A good example of how to use the *Remove appended node(s) from their current location* option is the sorting of nodes. Suppose we have a tree with this structure: **`$XML1/products/product/@name`**. We want to sort the **`product`** nodes on the basis of their **`@name`** values. We can do this with the Append Node(s) definition shown in the screenshot below.



- 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

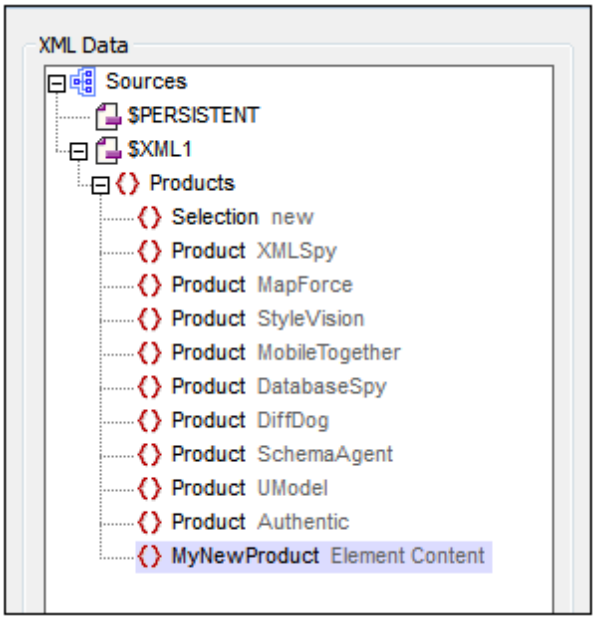
The node in the Append Node(s) definition that is targeted as the node in which to append child node/s, is automatically saved in MobileTogether Designer's built-in variable **`$MT_TargetNode`**. This variable can then be used in the second XPath expression of the definition, as has been done in the screenshot below.



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.

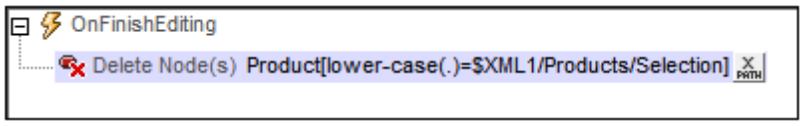
```
element {concat("MyNew", name($MT_TargetNode/*[last()]))} {"Element Content"}
```

The result of the Append Node(s) action defined above (when the target node is `$XML1/Products`) is shown in the screenshot below.

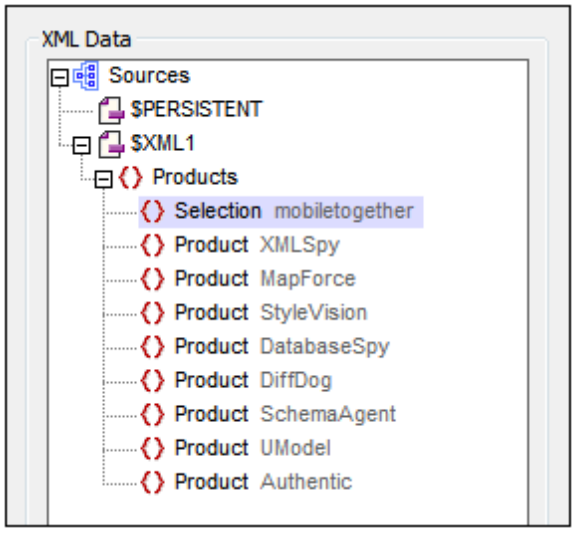

XML Data
Sources
$PERSISTENT
$XML1
Products
Selection new
Product XMLSpy
Product MapForce
Product StyleVision
Product MobileTogether
Product DatabaseSpy
Product DiffDog
Product SchemaAgent
Product UModel
Product Authentic
MyNewProduct Element Content

Delete Node(s)

The Delete Node(s) action deletes the node/s selected in the action's XPath expression.



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.



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) and Append Node(s).

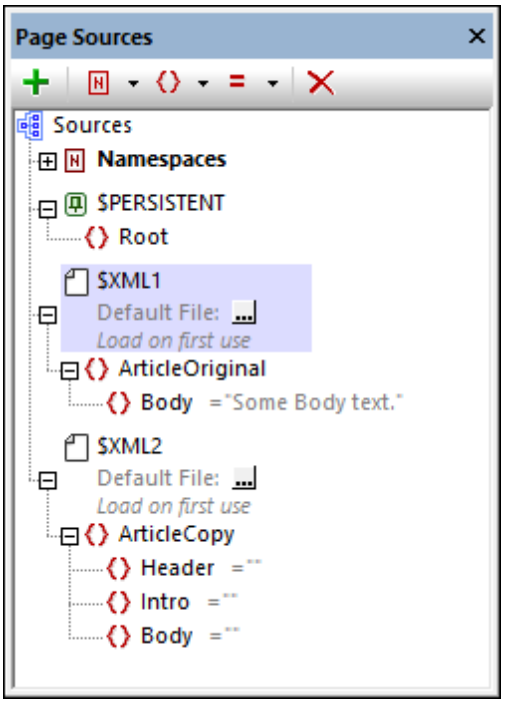
Examples

Given below are two examples that demonstrate how the Replace Node(s) action works.

Simple replacement

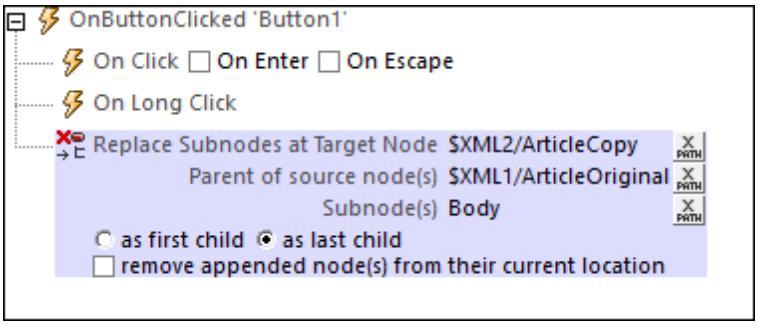
Consider the structure of the two XML trees shown in the Page Sources pane shown below.

- **`$XML1/ArticleOriginal`** has a child element named **`Body`**.
- **`$XML2/ArticleCopy`** has child elements named **`Header`**, **`Intro`**, and **`Body`**, in that order



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.



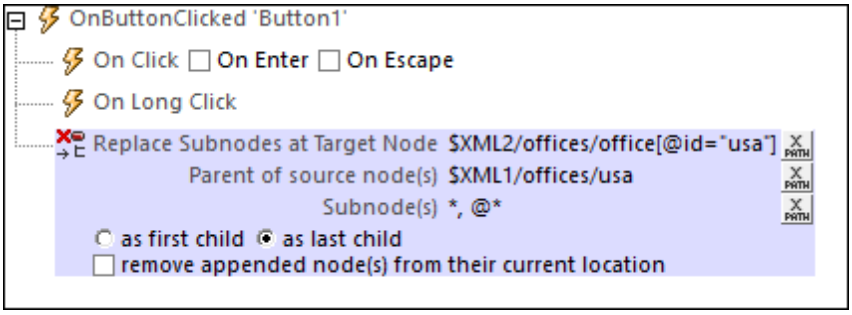
The action append a *subnode* named **Body** as a last child from a *source node* named **ArticleOriginal** to a target named **ArticleCopy**. The screenshot below shows the page sources during a simulation, after the action has been executed. Notice that the **Body** element of **ArticleCopy** has been replaced by the **Body** element of **ArticleOriginal**.

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"]**.



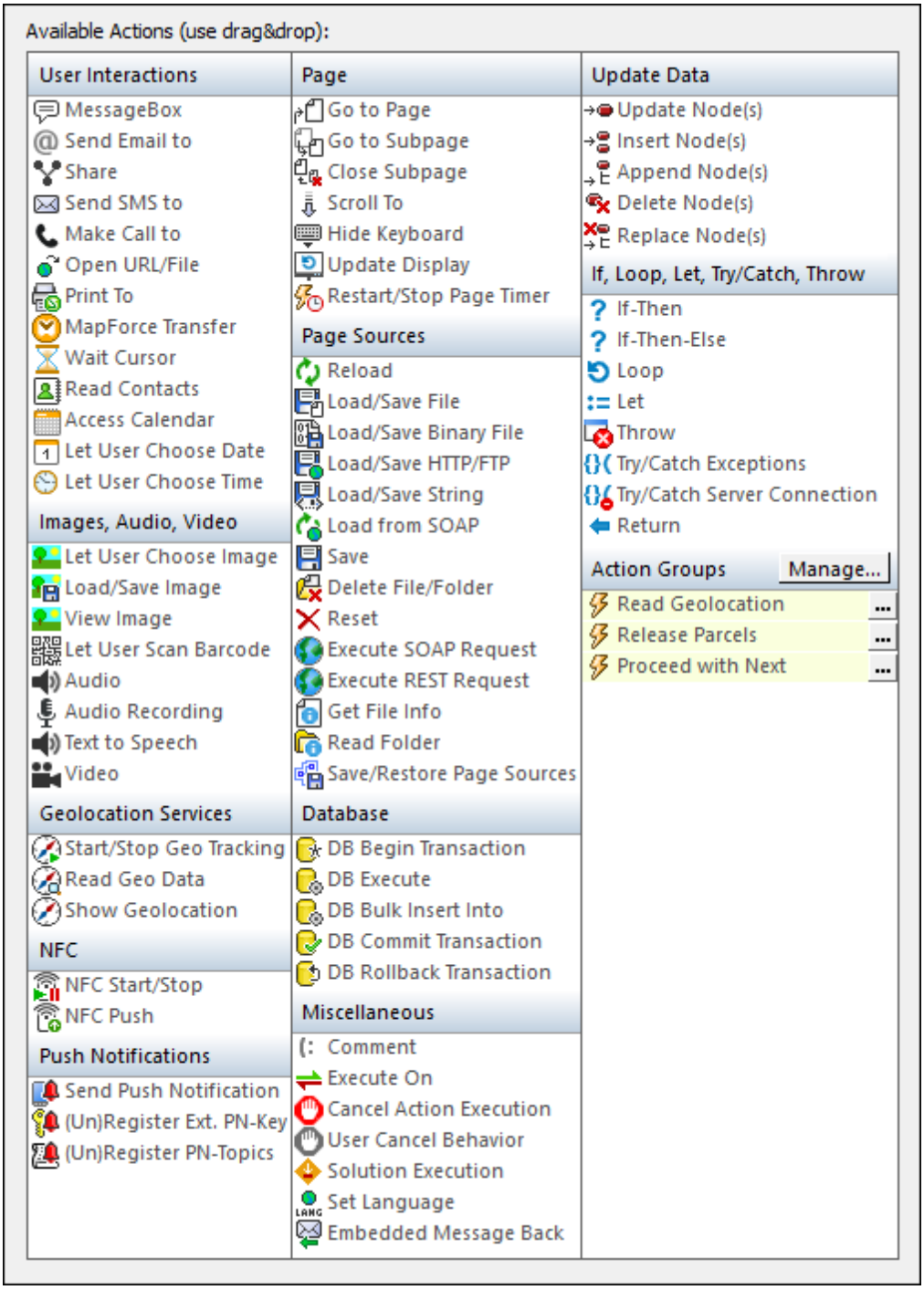
The exact sequence of actions is as follows:

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.

10.11 If, Loop, Let, Try/Catch, Throw

The following actions are available in the If, Loop group of the Actions dialog (*screenshot below*):

- If-Then
- If-Then-Else
- Loop
- Let
- Throw
- Try/Catch Exceptions
- Try/Catch Server Connection Errors
- Return



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 and Control Events*.

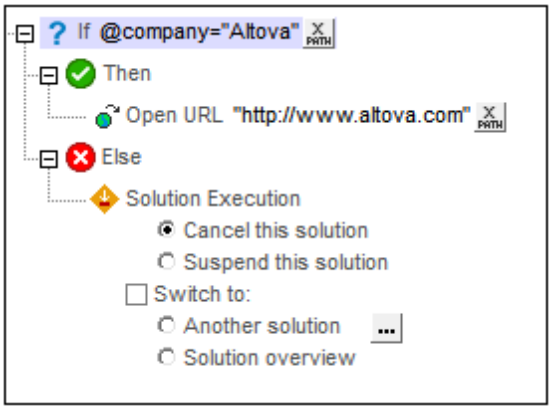
If-Then

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.

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-Then-Else

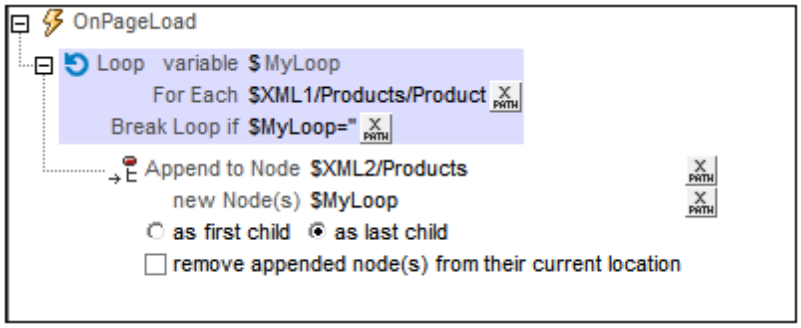
Sets the actions to be carried out for both the `true` and `false` evaluations of the `IF` condition. The action to be carried out for a `true` evaluation is dropped as a child of the `THEN` clause. The action to be carried out for a `false` evaluation is dropped as a child of the `ELSE` clause.



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.

Loop

The Loop action (*see screenshot below*) iterates over a sequence of items that you define using the *For Each* and *Break Loop If* settings. 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 action is carried out.

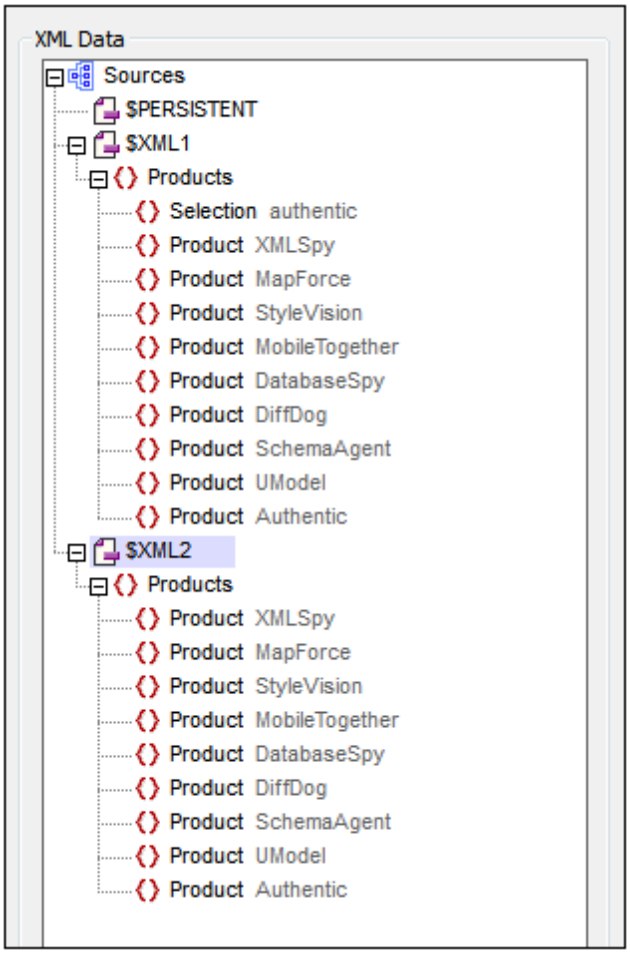


The sequence over which the loop iterates is defined by the XPath expressions of the *For Each* and *Break Loop If* settings. The key points to note are given below.

- *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*). The *For Each* setting can be specified with or without the *Break Loop If* setting. If there is no *Break Loop If* setting, 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 (**$MyLoop**). The variable will be in scope within the loop; this means that you cannot reference the variable in an XPath expression that is outside the loop. In the screenshot above, the loop variable is referenced in the *New Node* setting of the Append to Node action. This is a valid reference since the Append to Node action has been created within the loop; the variable is therefore in scope at this point. (It is also referenced in the *Break Loop If* setting.)
- *Break Loop If:* This XPath expression is evaluated before each iteration. If the expression evaluates to `true()`, then the loop is exited. In the screenshot above, the XPath expression `$MyLoop=''` specifies that the loop will be exited when the content of the `$MyLoop` variable is evaluated as being empty. This will happen when the first empty `Product` element is encountered and placed in the `MyLoop` variable.
- ***Important:*** Modifications to the sequence being iterated are not allowed while the loop is being executed. In the screenshot above, the new nodes that are added with the Append to Node action are added to another XML tree (`$XML2`). The sequence being iterated is in the `$XML1` tree, and cannot be modified. However, there is an elegant way to modify the sequence being iterated over, which is entirely consistent with XPath logic: Instead of iterating directly over the nodes, iterate over a number sequence that is tied to the node sequence. For example, instead of iterating over the sequence of `Product` nodes in the screenshot example above, we can iterate over a range of numbers that is tied to the node sequence. The XPath expression of the *For Each* setting can be changed from **`$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. As a result,

`Product` nodes are free to be modified within the loop. The current `Product` node within the loop can be accessed with the XPath expression: **`$XML1/Products/Product[$i]`**.

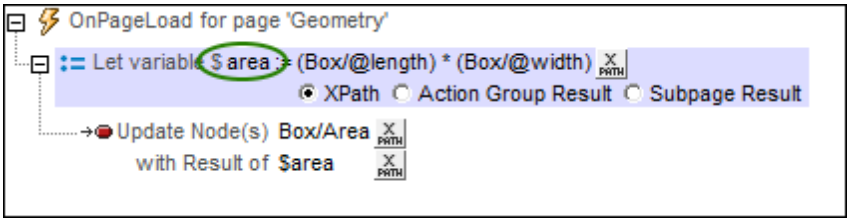
The actions defined in the screenshot above create a duplicate tree fragment. When the page is loaded, the Loop action iterates over the `$XML1/Products/Product` elements. During each iteration, the current `Product` node is stored in the `MyLoop` variable. This `Product` node (in the `$MyLoop` variable) is then added as the last child of the `$XML2/Products` node. The loop continues till the last `Product` element has been copied from `$XML1/Products` to `$XML2/Products`. *See screenshot below.*



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.

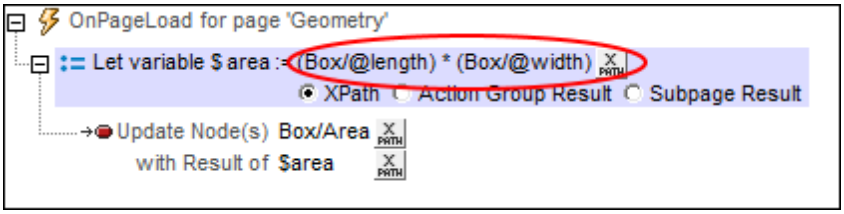
- To enter the name of the variable, double-click to the right of the **$** sign and type in the variable's name (*circled in green in screenshot below*).
- Select whether you wish to set the value of the variable via an XPath expression, an Action Group Result, or a Subpage Result.
- Define one or more child actions of the Let action. For example, the Let action shown in the screenshot below contains an Update Node(s) action, which updates a node with the value of the variable defined in the Let action.



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.

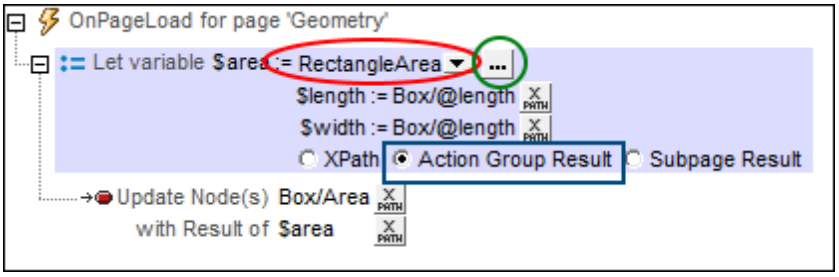
The XPath option

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.

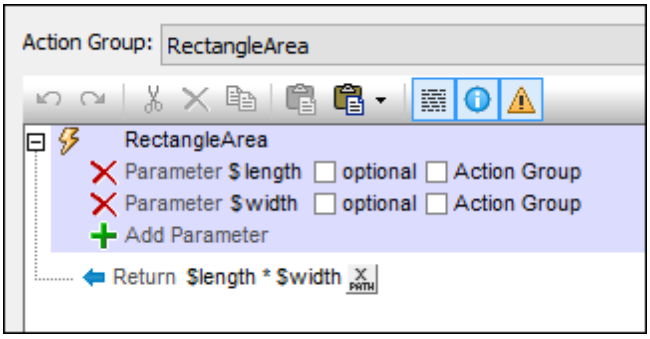


The Action Group Result option

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.



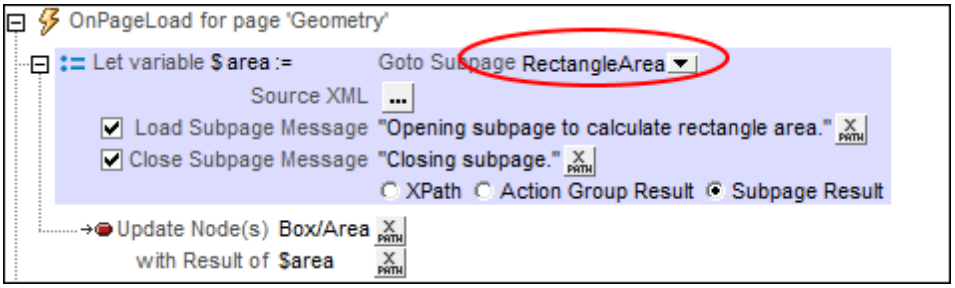
The Action Group returns a result via the Return action (*see screenshot below*). In the screenshot below, for example, we declare two parameters (`$length` and `$width`); in the Return action, we multiply the values of the two parameters. Note that the values of the parameters are obtained at runtime 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.



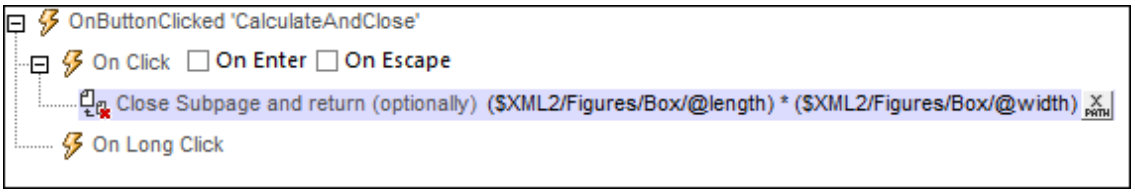
The Subpage Result option

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 action, and are described in detail there.

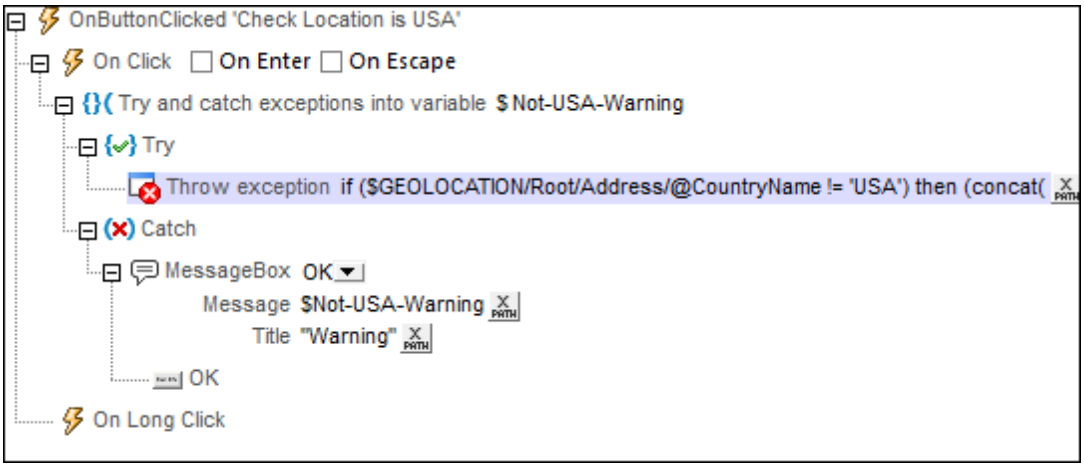


At runtime, when the Let action is executed, the subpage is opened and it will be processed as defined in its design. The subpage will be closed when the Close Subpage action is executed. This action has an optional return value which is calculated by an XPath expression (*see screenshot below*). This returned value will be passed to the Let action and will become the value of the variable defined in the Let action.



Throw

The Throw action is intended to be used in the Try part of a Try/Catch action (*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; *in the screenshot below, this variable is named* `$Not-USA-Warning`.



In the example shown in the screenshot above, we throw an exception if the geolocation of the device 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 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 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 shows how the Try/Catch and Throw actions can be used.

Try/Catch Exceptions

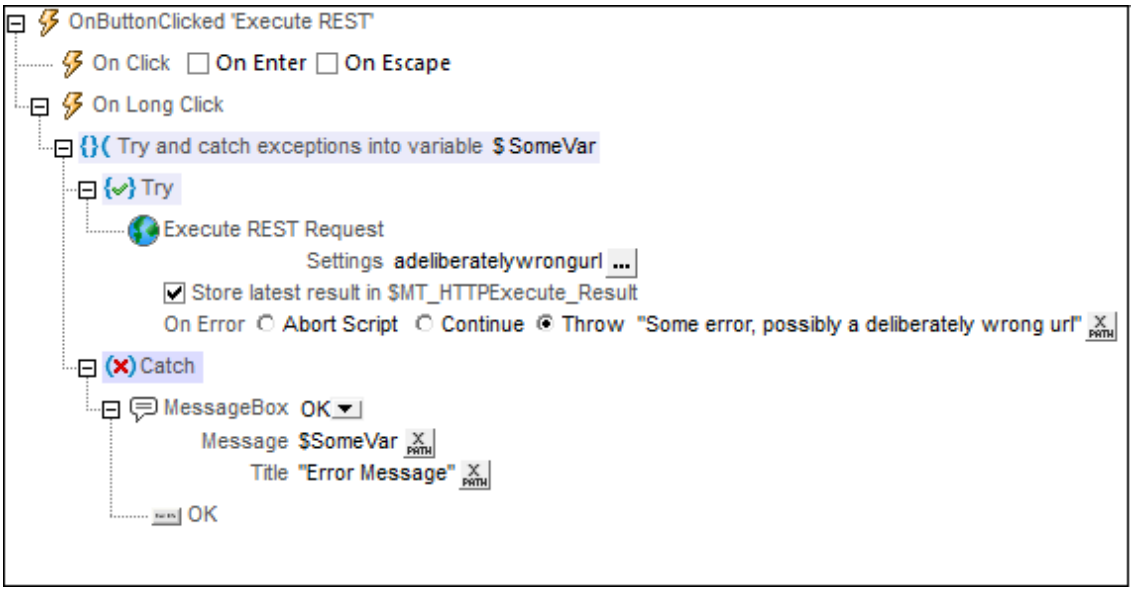
The Try/Catch Exceptions action has two parts (*highlighted in the screenshot below*):

- *Try:* Defines a condition or an action to try.
  - A condition is defined in the XPath expression of a Throw action. (*See the Sharing Geolocations tutorial for an example of using the Throw action*.)
  - 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.)
  - Both the Throw action (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 (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.



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.

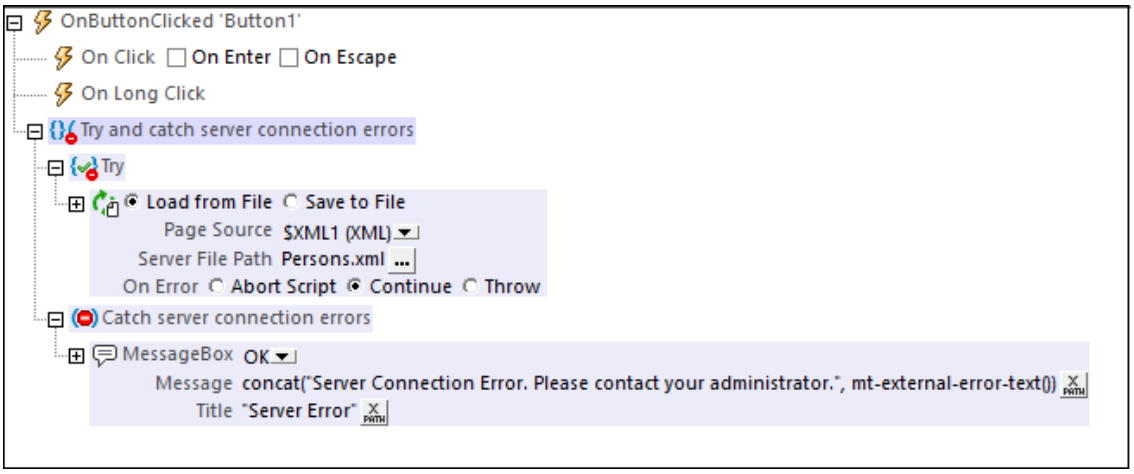
3. In the Try part of the action, we have selected the *Throw* option for the *On Error* sub-action of the Execute REST request, and entered an exception message as the option's XPath expression. As a result, if an error is detected, an exception is thrown and the exception message is stored in the **$SomeVar** variable.
4. In the Catch part of the action, we have defined a Message Box action to show the message that is stored in the **$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 action in the *Continue* option's *On Error* sub-action.

The tutorial Sharing Geolocations shows how the Try/Catch and Throw actions can be used.

Try/Catch Server Connection Errors

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*).

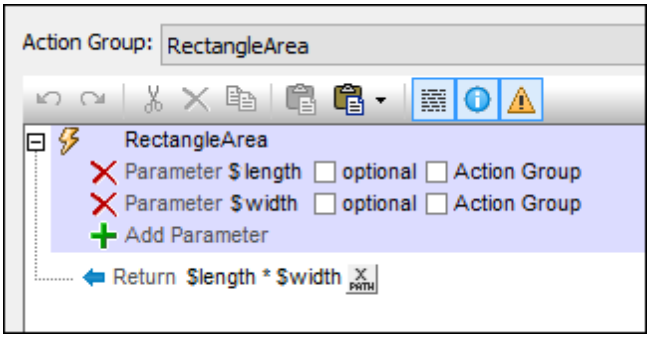


Note: The MobileTogether XPath extension function **`mt-external-error-text()`** can be used in the *Catch* part to display information about the server connection error.

Note: When a server connection error occurs, the first of the following actions that exists is triggered: (i) a Try/Catch Server Connection Errors action, (ii) action/s for the **`OnServerConnectionError`** event, (iii) a MobileTogether message about the error, following which the workflow is resumed.

Return

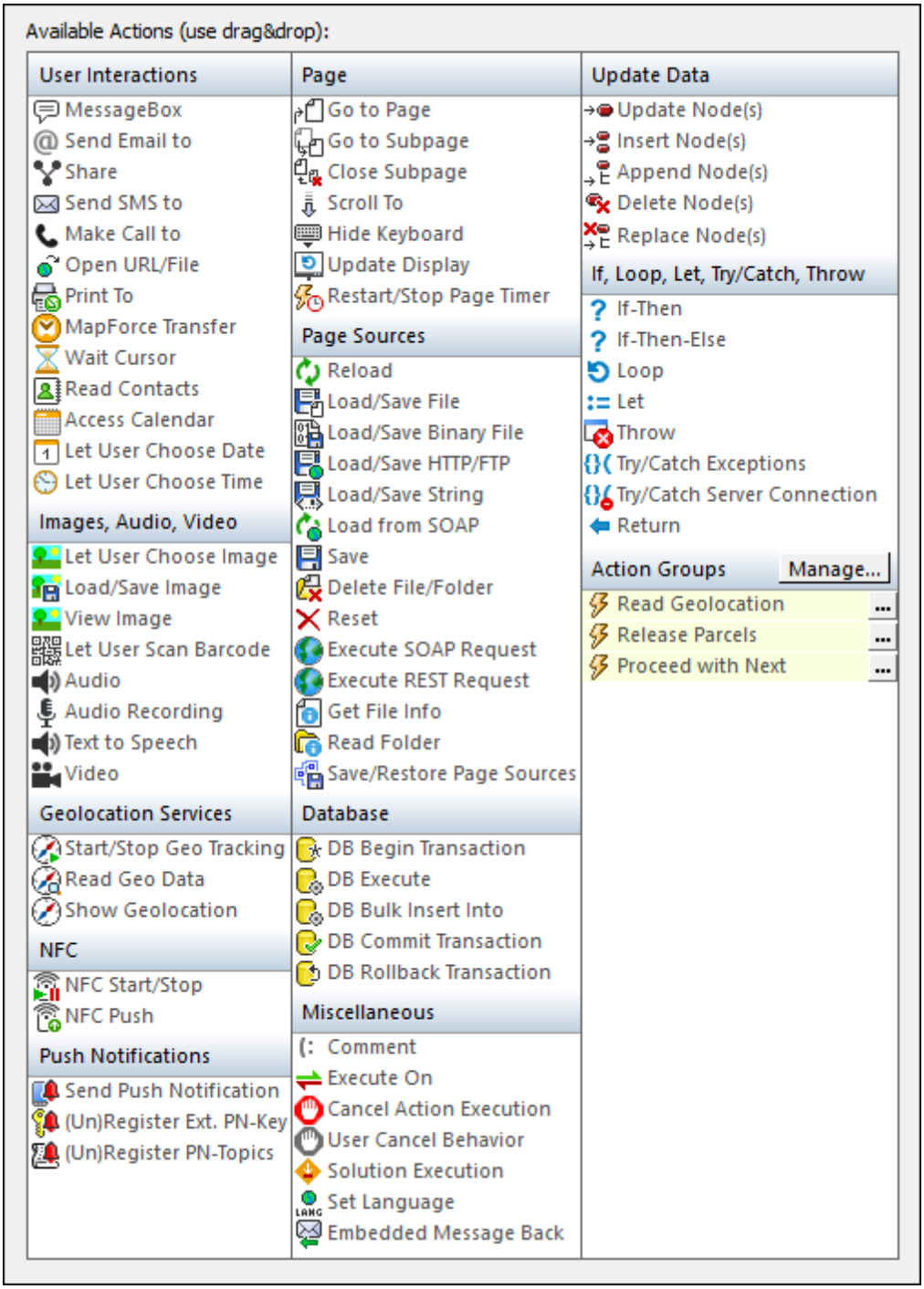
In a Return action, an XPath expression defines the value that is returned by an Action Group. This value is called the Action Group Result, and it can be used to set the value of a variable defined in a Let action. In the screenshot below, for example, the Return action defines the value that is returned 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 that calls the Action Group. See the section *Setting Variable Values via Action Groups* for a description and example of how the Return action can be used.



10.12 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 more efficient to create an Action Group that contains this sequence of actions and then execute this Action Group at the various workflow points. Action Groups can also be parameterized.

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, it is available for use on any page event or control event of the project. You can use an Action Group as many times as you like across events.



This section describes:

- How to add, delete, and edit Action Groups
- How to use Action Groups for reusing actions
- How to use parameters in Action Groups
- How Action Groups can themselves be used as values of parameters in Action Groups
- How Action Groups can be used to provide the value of a variable defined with the Let

action.

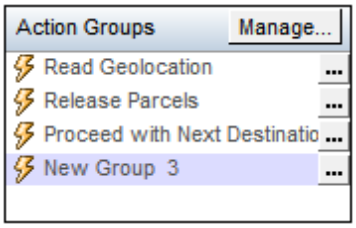
Managing Action Groups

An Action Group 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, and they are managed in the Manage Group Actions dialog, which is described below.

Accessing the Manage Group Actions dialog

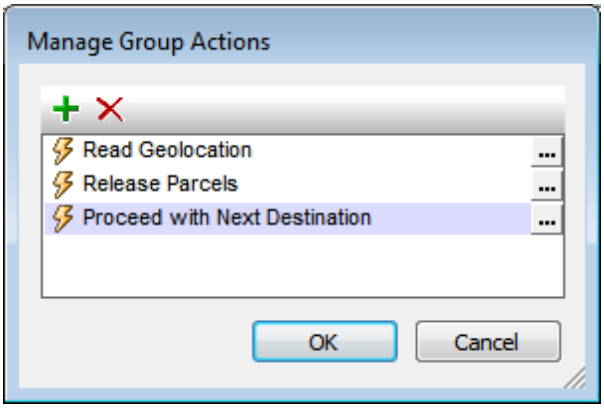
To access the Manage Group Actions dialog, do the following:

1. Open the Actions dialog: (i) via either a control event or page event, or (ii) by clicking **Project | Action Groups**.
2. In the Actions dialog, click **Manage** in the Action Groups pane (*see screenshot below*).



The Manage Group Actions dialog

In the Manage Group Actions dialog (*screenshot below*), you can add and delete Action Groups, and access the Action Group to edit it.



Adding an Action Group

Click **Add** in the toolbar of the dialog. A new Action Group with a default name is added to the list in the dialog (*see screenshot above*). Double-click the default name to edit it, and then click **OK**.

Deleting an Action Group

Select the Action Group you want to delete, then click **Delete** in the toolbar of the dialog.

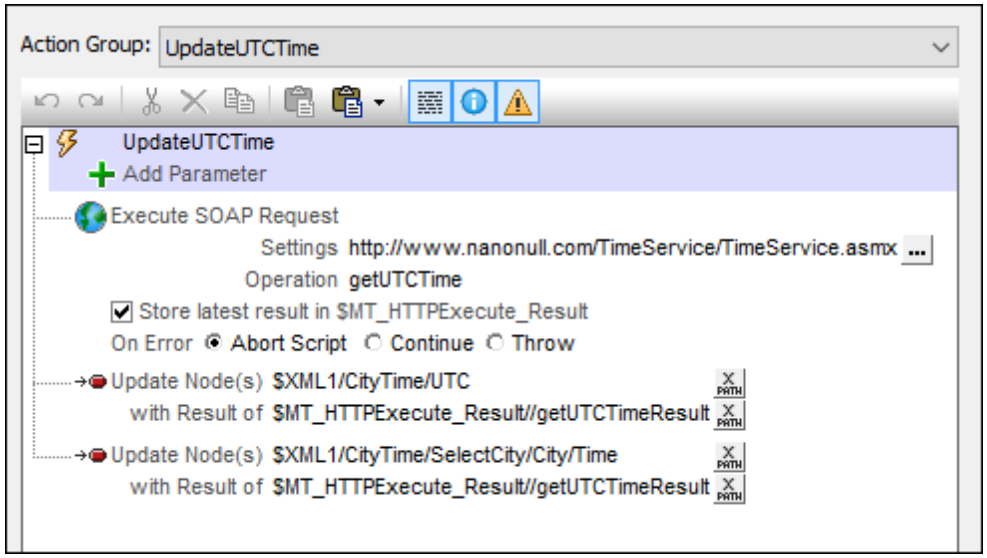
Accessing an Action Group to edit it
Access an Action Group in one of the following ways:

- In the Manage Group Actions dialog, click the **Edit** button of the Action Group you want to edit.
- In the Action Groups pane of the Actions dialog (*screenshot below*), click the **Edit** button of the Action Group you want to edit. Alternatively, you can double-click the Action Group entry.

Action Groups for Reusing Actions

Action Groups can be used to group a set of actions for reuse. Set up the an Action Group for this purpose as follows:

1. Access the Action Group in order to edit it.
2. Drag and drop actions and/or action groups from the Available Actions pane on the left, and define them. For example, in the screenshot below, we have defined three actions for the *UpdateUTCTime* Action Group.



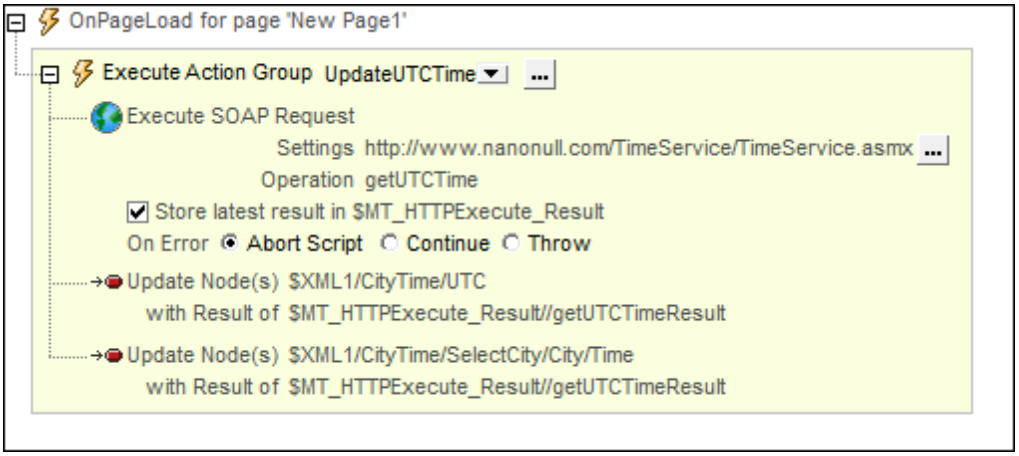
3. Click **OK** to finish.

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.



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.
- You can click the Action Group's **Edit** button to edit the currently selected Action Group.

Action Groups with Parameters

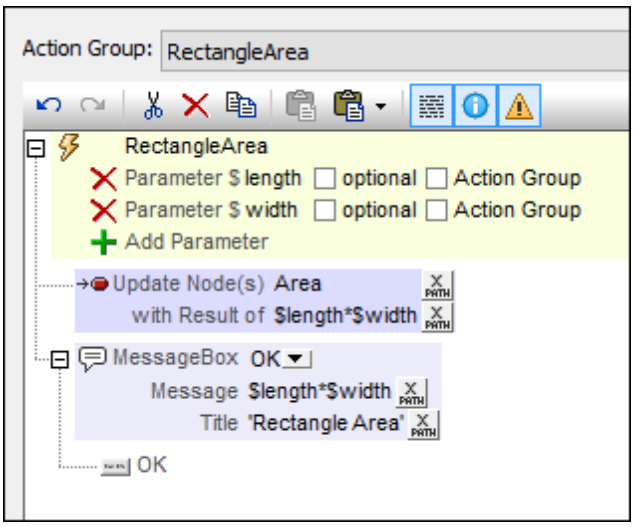
Parameters in Action Groups work as follows:

- 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, values for both these parameters must be submitted; otherwise an error is reported.



The Action Group consists of two actions:

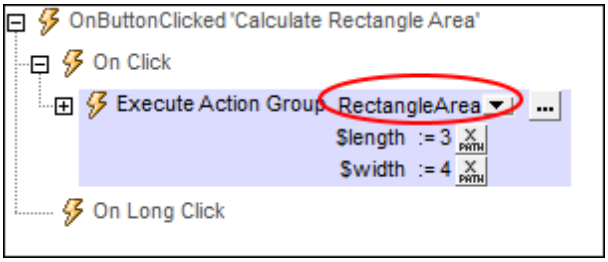
- An Update Node action, which multiplies the two parameters **`$length`** and **`$width`** to generate a value that updates the `Area` node.
- A Message Box action, which displays a message box containing the value obtained by multiplying the two parameters **`$length`** and **`$width`**.

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.

Supplying parameter values

The values of an Action Group's parameters are supplied to the Action Group via the Execute Action Group action. The Execute Action Group action is defined for an event, and it it is processed when that event is triggered. So, when a given event is triggered, parameter values can be passed to an Action Group. The Action Group then uses these values when carrying out the actions defined in it

To create an Execute Action Group action, drag and drop the relevant Action Group (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. 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*).



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:

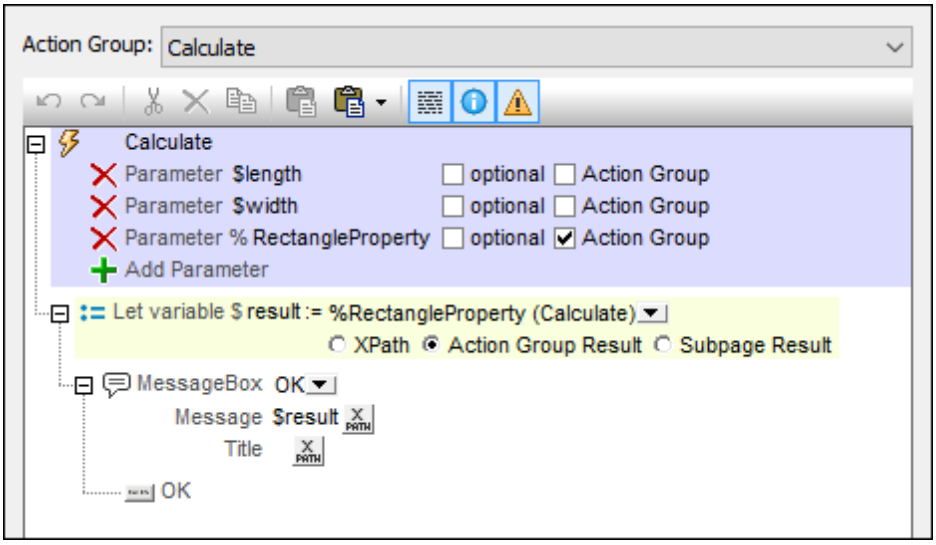
1. When the event is triggered, the parameter values (as specified in the Execute Action Group action) are passed to the respective parameters in the Action Group.
2. The Action Group's actions are processed. Where these use the Action Group's declared parameters, the supplied parameter values are substituted.

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. They are indicated in MobileTogether Designer by the dollar symbol **$** in front of their names.
- *Action-Group parameters*, which take Action Groups as their values. They are indicated in MobileTogether Designer 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 and then check its *Action Group* check box (*see screenshot below*).



How Action-Group parameters work

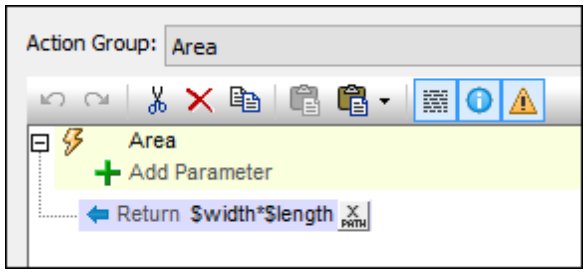
An Action-Group parameter takes as its value an Action Group that declares no parameter. This enables the called Action Group to be treated like a function that performs MobileTogether tasks. Action-Group parameters work in a broadly similar way to simple parameters.

- You declare Action-Group parameters in an Action Group. The containing Action Group can then define actions that use these Action-Group parameters. Alternatively, the containing Action Group can directly process Action-Group parameters via the Execute Action Group action (see the section *Processing Action-Group parameters* below).
- The values of Action-Group parameters are supplied via the Execute Action Group action that calls the containing Action Group. Allowed values are those Action Groups (in the project) that have no parameter.

Usage example

Here is a simple example to show how Action-Group parameters can be used. We create four Action Groups to calculate and display three properties of rectangles (area, diagonals, and perimeter):

- **Calculate** (*see screenshot above*): Declares two simple parameters (**$length** and **$width**) and an Action-Group parameter (**%RectangleProperty**). The Action-Group parameter 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. This action defines a variable called **$result** to take the Action Group Result of the Action Group that is the value of the **%RectangleProperty** parameter. Next comes a Message Box action to display the value of the **$result** variable that was defined in the previous action.
- **Area** (*screenshot below*): This Action Group contains a single Return action 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**. If it is, then, when **%RectangleProperty** is processed, the **$result** variable in the **Calculate** Action Group will receive the return value of the **Area** Action Group.



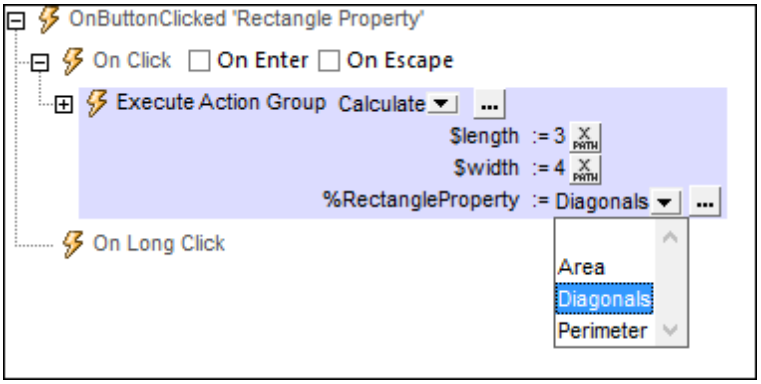
- **Diagonals**: This Action Group is similar to the **Area** Action Group. It contains a single Return action that returns the length of the diagonals of a rectangle (which is the square root of the sum of the squares of the two simple parameters **$length** and **$width**; *XPath:* `math:sqrt($width*$width + $length*$length)`). If this Action Group is the value of **%RectangleProperty**, then the value returned by it is set as the value of the **$result** variable in the **Calculate** Action Group.
- **Perimeter**: This Action Group is similar to the **Area** and **Diagonals** Action Groups. It contains a single Return action that returns the perimeter of a rectangle (*XPath:* `2*$width + 2*$length`). If this Action Group is the value of **%RectangleProperty**, then the value returned by it is set as the value of the **$result** variable in the **Calculate** Action Group.

In the description of the four Action Groups above, note the following:

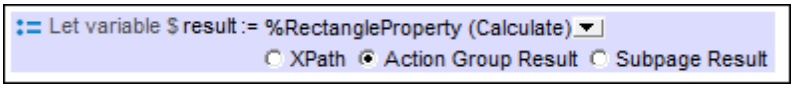
- The **Calculate** Action Group contains three parameters: two simple parameters and one Action-Group parameter.
- The other three Action Groups (**Area**, **Diagonals**, **Perimeter**) declare no parameter.
- Each of the other three Action Groups (**Area**, **Diagonals**, **Perimeter**) has a Return action that calculates a value from the simple parameters declared in **Calculate**. The returned value is the Action Group Result of each of these Action Groups.
- 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.
- The Action-Group parameter **%RectangleProperty** is processed in the **Calculate** Action

Group. It provides the value of the **$result** variable, which is displayed in a message box (*see screenshot* of **Calculate** Action Group above).

Supplying the values of Action-Group parameters
Since the parameters are declared in the **Calculate** Action Group, their values are passed through when the **Calculate** Action Group is called (via an Execute Action Group action). So we could, for example, define an `OnButtonClicked` event as shown in the screenshot below.



We create an Execute Action Group action (by dragging the **Calculate** Action Group from the Action Groups pane) into the event's pane. The parameters of the **Calculate** Action Group are automatically displayed. For the **%RectangleProperty** Action-Group parameter, we can select the Action Group that we want to be the value of the parameter (*see screenshot above*). So when the **Calculate** Action Group is executed, it will be passed the three parameter values specified in the call to it. When the Let action in the **Calculate** Action Group is processed (*see screenshot below*), the **$result** variable will be set to the Action Group Result of the Action Group that was specified as the value of **%RectangleProperty**.



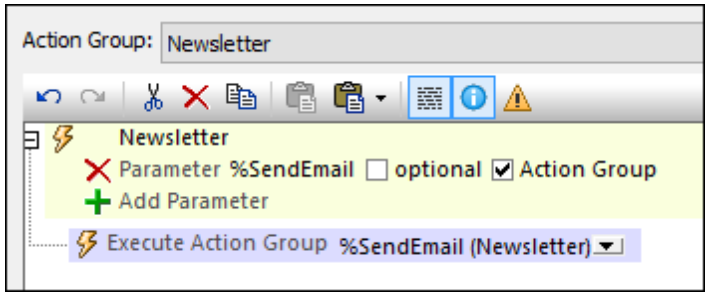
Note: The example above is deliberately simple, and aims to show the mechanism behind Action-Group parameters. But Action-Group parameters are best used with dynamic content and to execute complex actions.

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—which can then be used to set the value of a variable that is defined with a Let action. 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*

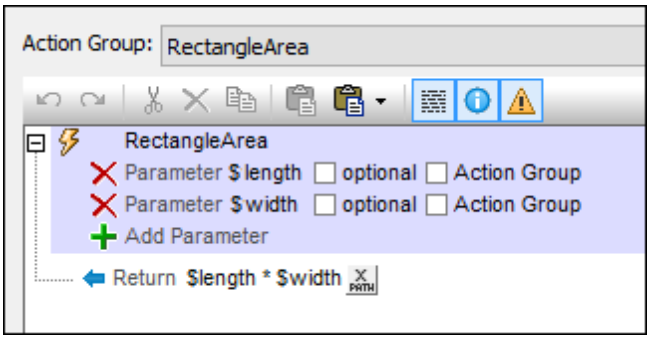
screenshot below). This is done by dragging the Action-Group parameter from the Action Groups pane into the event's definition.



You could of course combine both sets of actions.

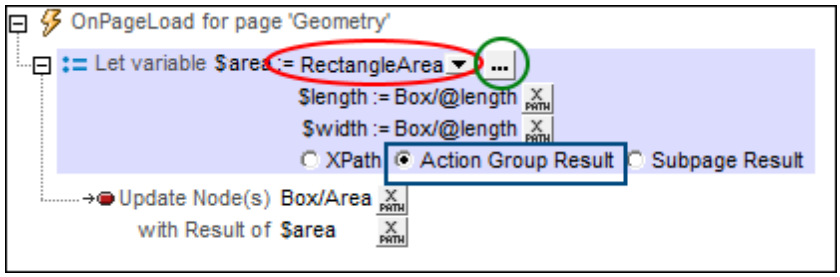
Variables and Action Group Results

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 to produce the Action Group Result.) When a Let action is defined, we can assign the Action Group Result to a variable and use the variable in child actions of the Let action.



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 **`$width`**. Note that: (i) the parameters have been declared but no values have been defined for them; (ii) the parameters are in scope only within the Action Group, and cannot be used outside it. If a parameter is defined as optional, then it is not an error if it is not used in the definition of the variable (*see Step 3 below*).
2. Add a Return action. Use an XPath expression to define the result to return. This result will be the Action Group Result that the Let action can use. In the screenshot above, we have defined an expression that multiplies the values of the **`$length`** and **`$width`** parameters. Note that the Return action is within the Action Group. Consequently, the parameters are in scope.
3. To declare that a variable defined by a Let action will have the value of an Action Group Result, define the Let action as follows (*see screenshot below*): (i) Double-click to the right of the variable's **$** 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.



In the screenshot above, we have given, in the Let action, 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 **$width** parameters (which were declared in the `RectangleArea` Action Group). When the Let action is executed, the parameter values are passed to the Action Group, where the Return action uses the parameter values in its calculation. The result is returned and becomes the value of the variable defined in the Let action. In our example above, the values of the **$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.

4. The variable defined in the Let action can now be used in child actions of the Let action. 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 for more information.

Chapter 11

Design Objects/Features

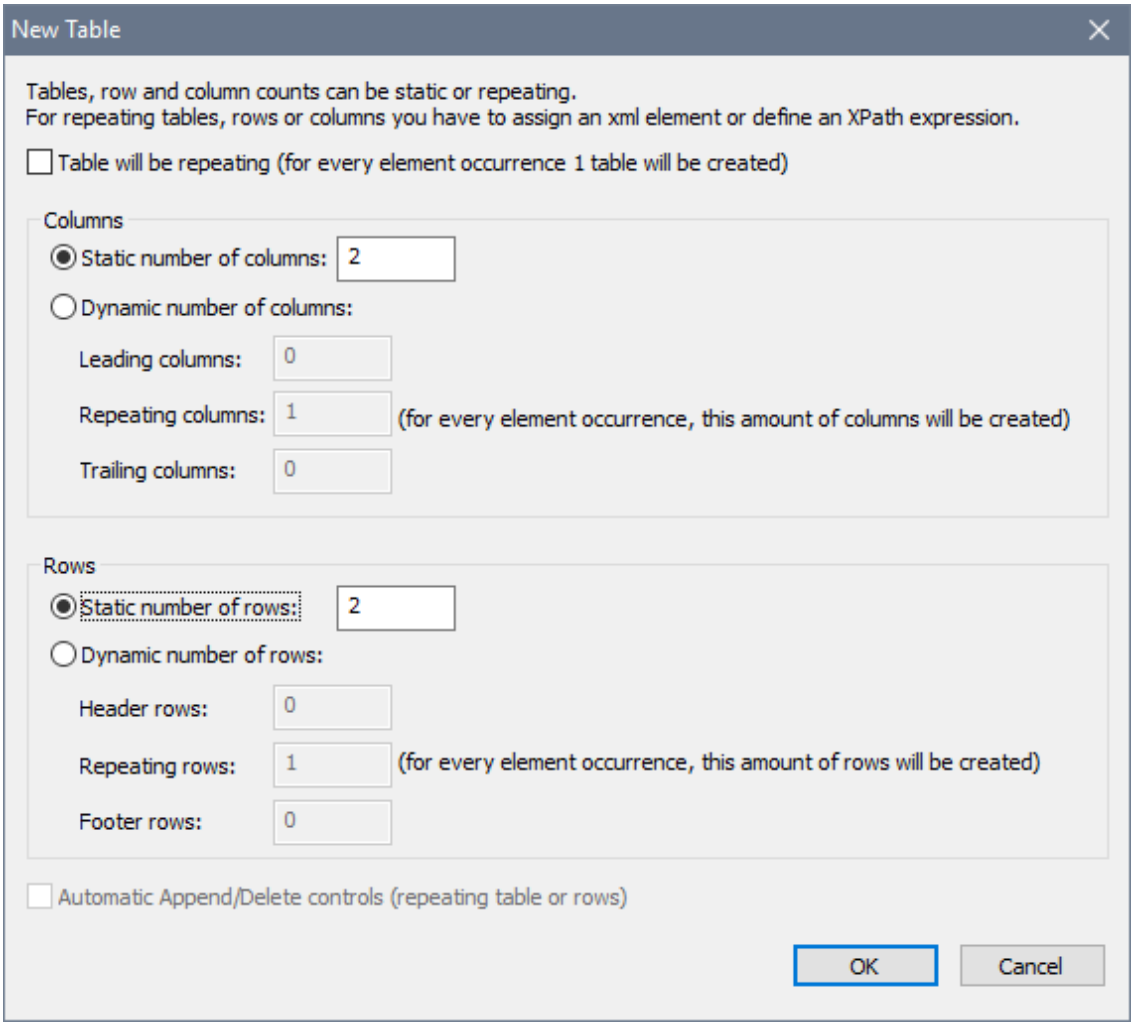
11 Design Objects/Features

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, which explains how to use the different types of table that can be used in a design
- Images, which describes the powerful image handling capabilities of MobileTogether
- Audio, Video, which provides an overview of the audio-video features of MobileTogether
- NFC, which describes how NFC messages can be sent and received, and set up to be processed further
- Push Notifications, which describes how to set up the sending and receiving of push notifications in solutions
- Charts, which explains how charts can be configured in the design
- Style Sheets, which describes how global styles can be applied at the project, page, table, and control level
- Rich Text, which describes how the Rich Text control and Rich Text style sheets are used to display and edit rich text in a solution.
- Hyperlinking to Solutions, which describes mechanisms for starting a solution from (i) another solution, or (ii) a link in an email

11.1 Tables

You can insert tables into your design by dropping the Table control at the location where you want the table. For each Table control, 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.



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, which are useful for presenting data in neatly ordered rows and columns. In the New Table dialog, you can specify the number of rows and columns. You can modify the structure at any later time.

- Repeating tables, in which every occurrence of the associated data-source element is created as a row, and where each row consists of the entire table as created in the design. Repeating tables can contain dynamic (repeating) columns (*see fourth point of this list*).
- Tables with dynamic rows, in which, just as with repeating tables, every occurrence of the associated data-source element is created as a row. The difference from repeating tables is that each element occurrence is created as a row rather than as a table. Dynamic rows can be comprise of dynamic (repeating) columns.
- Tables with dynamic columns, which can be of two kinds: (i) where the rows are static, and (ii) where the rows are dynamic. The dynamic row and dynamic column is each respectively associated with a different repeating element from the data source.

Table structure and properties

The internal structure of the table can be modified at any subsequent time. Properties for the entire table and for individual columns, rows, and cells are specified by selecting the respective table component and assigning it properties in the Styles & Properties Pane.

For more information, see the sections: Table Properties and Table Context Menu. Also see the description of the Table control.

In this section

This section is organized as follows:

- Static Tables
- Repeating Tables
- Dynamic Rows
- Dynamic Columns
- Table Properties
- Table Context Menu

Static Tables

Static tables 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 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.

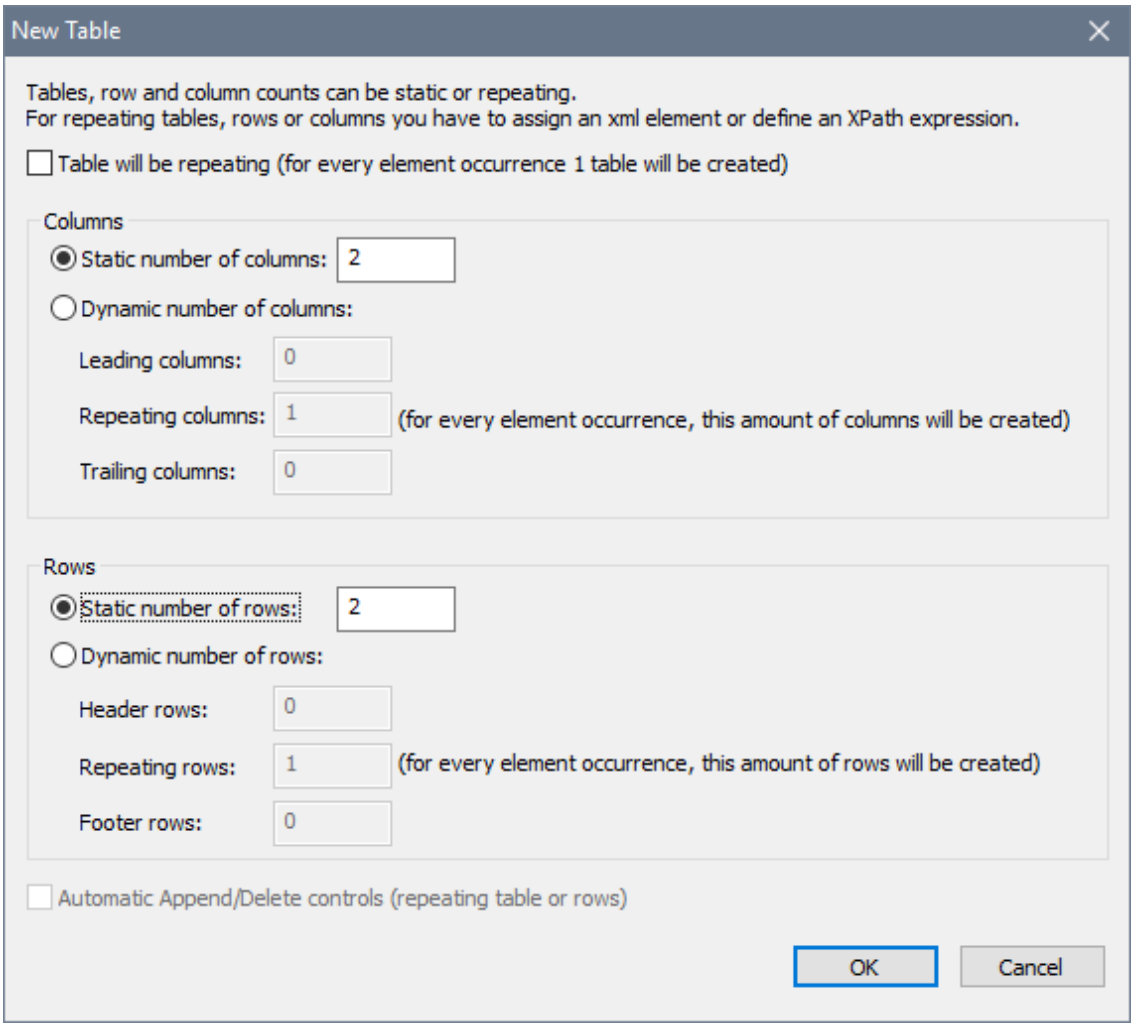


Table restructuring commands are available in the context menu of the table.

Table formatting properties are available in the Styles & Properties Pane.

Table contents

The cells of a static table can contain the following:

- Static text
- A node from a data source
- A page control (with or without a link to a data source node)
- A nested table

Cell content is typically a page control. 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 each.

- The two cells of the first table contain combo boxes.
- The second table contains two empty cells (used for spacing) and six cells with one Label control each.
- The third table contains two buttons. (The lightning symbols on some controls indicate that the controls have actions defined for them.)

Repeating Tables

Repeating tables work as follows:

- A repeating element in the data source is associated with the repeating table (*see examples below*).
- The table can be designed to contain any number of rows and columns.
- For each occurrence of the repeating element, the table that you design will be repeated. So if the table design contains two rows, then a table with two rows will be created for each occurrence of the repeating element.
- The content of the table can be dynamic. XPath expressions inside the table will have the associated repeating element as its context node. For every occurrence of the element, the specific occurrence will be the context node for that specific table (*see examples below*).
- An Append/Delete control can be added to the table that enables end users to add new rows and delete individual rows. For example, if the user adds a row, a new occurrence of the associated table element will be added to the tree of the data source. These modifications can be saved back to the data source, thus enabling end users to modify the data source.

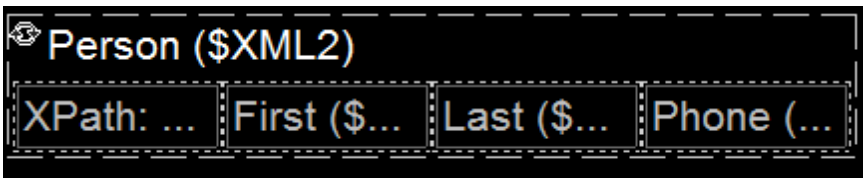
A repeating table is defined at the time when the Table control is dropped into the design.

Example

A **Person** element in the data 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:

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 data source for use elsewhere in the document.

Difference between a repeating table and a table with dynamic rows

A *repeating table* is different than a *table with dynamic rows* in the following ways:

- In a *repeating table*, it is the **entire table** that is associated with the repeating data structure. A new table is generate for each occurrence
- In a *table with dynamic rows*, a **table row group** is associated with the repeating structure.

This difference has two design effects:

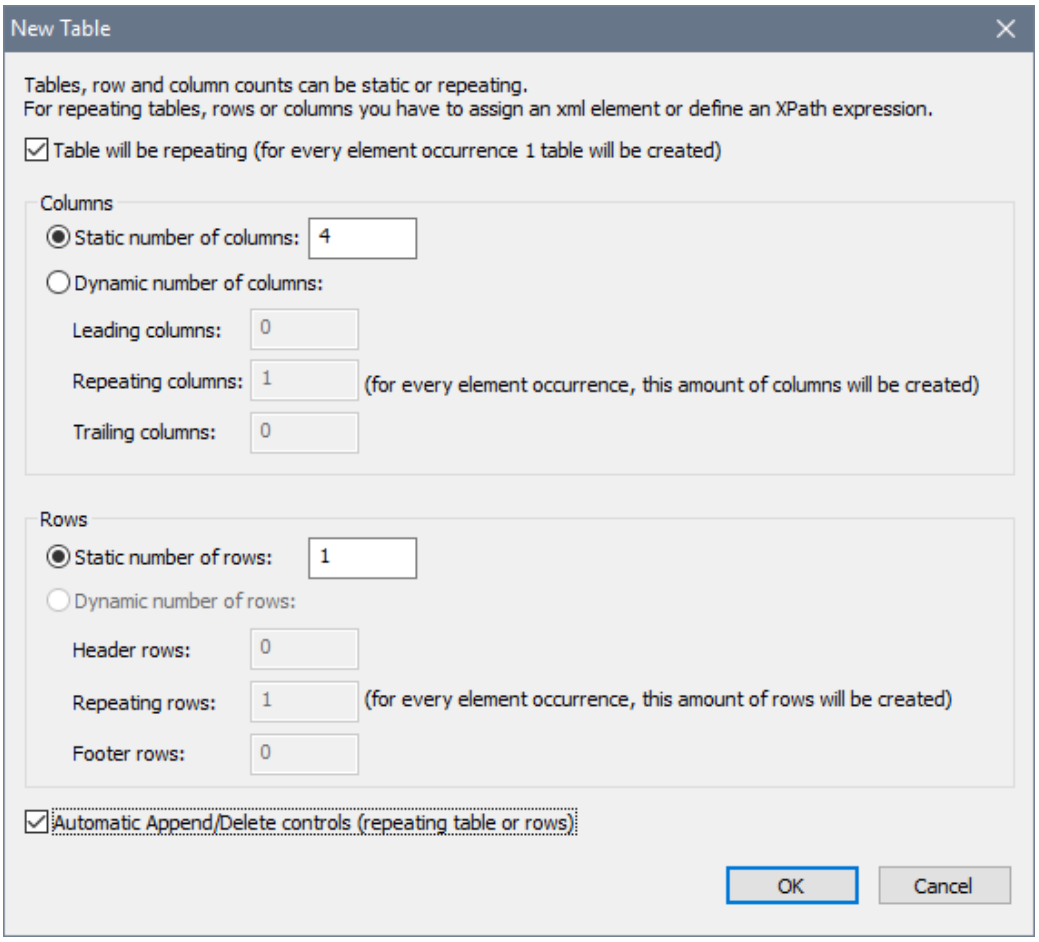
- A *table with dynamic rows* can have a header and/or footer that applies to the entire table. It is a header/footer for the table. If a header/footer is required for a *repeating table*, they can be manually added outside the repeating table. If added inside the *repeating table*, then these will repeat with each table for each element occurrence.
- Since tables are typically rendered with space above and below them in device displays, *repeating tables* will contain vertical space between its each pair of repeated tables.

To change a static table to a repeating table after the table has been created, change its Repeating property to `true`, then associate the table with a repeating element from a data source.

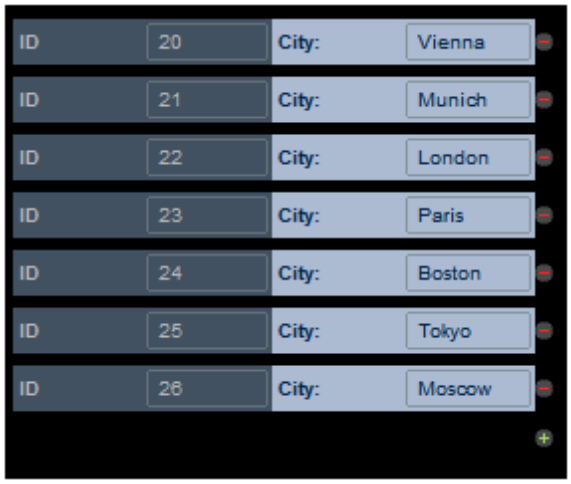
Creating a repeating table

Set up a repeating table as follows:

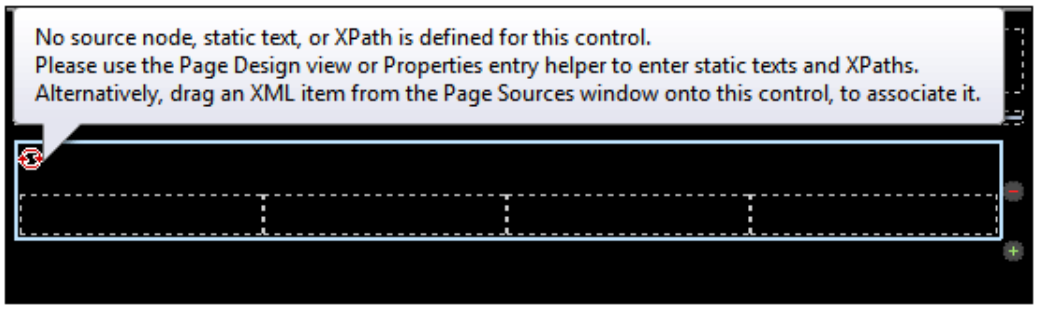
1. In the New Table dialog that appears when the Table control is dropped in the design (*screenshot below*), check *Table will be repeating* to make the table repeating. Note that it is the table that repeats for every instance of a repeating data row.



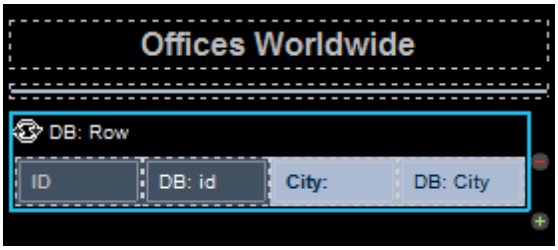
2. Specify the number of static columns and rows that the repeating table will have. You can subsequently add columns and rows to the repeating table via the table's context menu.
3. Specify whether automatic Append/Delete controls are to be added. If added, each repeating table in the solution, effectively a row, will have a **Delete** button, and the entire repeating table structure will have an **Append** button to append a repeating table (row) to the structure (*see the screenshot of a simulated solution below*).



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 data source (*see screenshot below*).



5. Associate a repeating element with the repeating table by dragging and dropping the element from the Page Sources Pane 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 data 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.



This repeating table produces the following repeating structure in the MobileTogether solution.

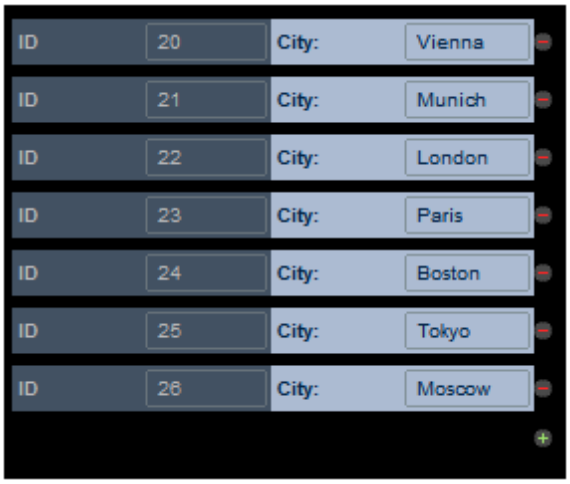


Table restructuring commands are available in the context menu of the table. Table formatting

properties are available in the Styles & Properties Pane. See also Dynamic Rows and Dynamic Columns.

Dynamic Rows

Dynamic rows work as follows:

- A dynamic row is associated with a repeating element in a data source.
- When the table is rendered, the number of rows in it will dynamically match the number of occurrences of the repeating element. Each table row will correspond to one occurrence of the repeating element.
- When you define the dynamic row, you can specify how many rows will be repeated for each element occurrence. So you could specify that each dynamic (or repeating) row group contains two rows. In this case, the entire row group of two rows is generated for each element occurrence.
- The context node for each dynamic row (or row group) will be the specific element occurrence.
- An Append/Delete control can be added to the table that enables end users to add new rows and delete individual rows. For example, if the user adds a row, a new occurrence of the associated row element will be added to the tree of the data source. These modifications can be saved back to the data source, thus enabling end users to modify the data source.

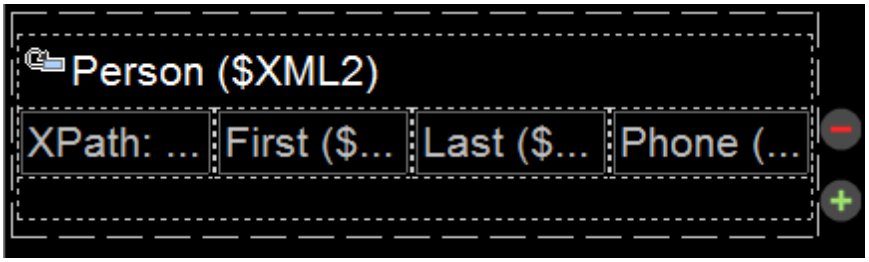
A table with dynamic rows is defined at the time when the Table control is dropped into the design.

Example

A **Person** element in the data source contains a repeating structure (say, of **First**, **Last**, and **Phone** elements). The **Person** element can occur multiple times (its occurrences). If a dynamic row (more precisely, a row group) is associated with the **Person** element, then the table will be generated with as many row(group)s as there are **Person** elements (*see table below*). If the number of **Person** elements in the data sources changes, then the number of rows in the table is modified automatically.

	<First>	<Last>	<Phone>
<Person>			
<Person>			
<Person>			

The design shown in the screenshot below contains a table that contains a single dynamic 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 data source for use elsewhere in the document.

Difference between a repeating table and a table with dynamic rows

A *repeating table* is different than a *table with dynamic rows* in the following ways:

- In a *repeating table*, it is the **entire table** that is associated with the repeating data structure. A new table is generate for each occurrence
- In a *table with dynamic rows*, a **table row group** is associated with the repeating structure.

This difference has two design effects:

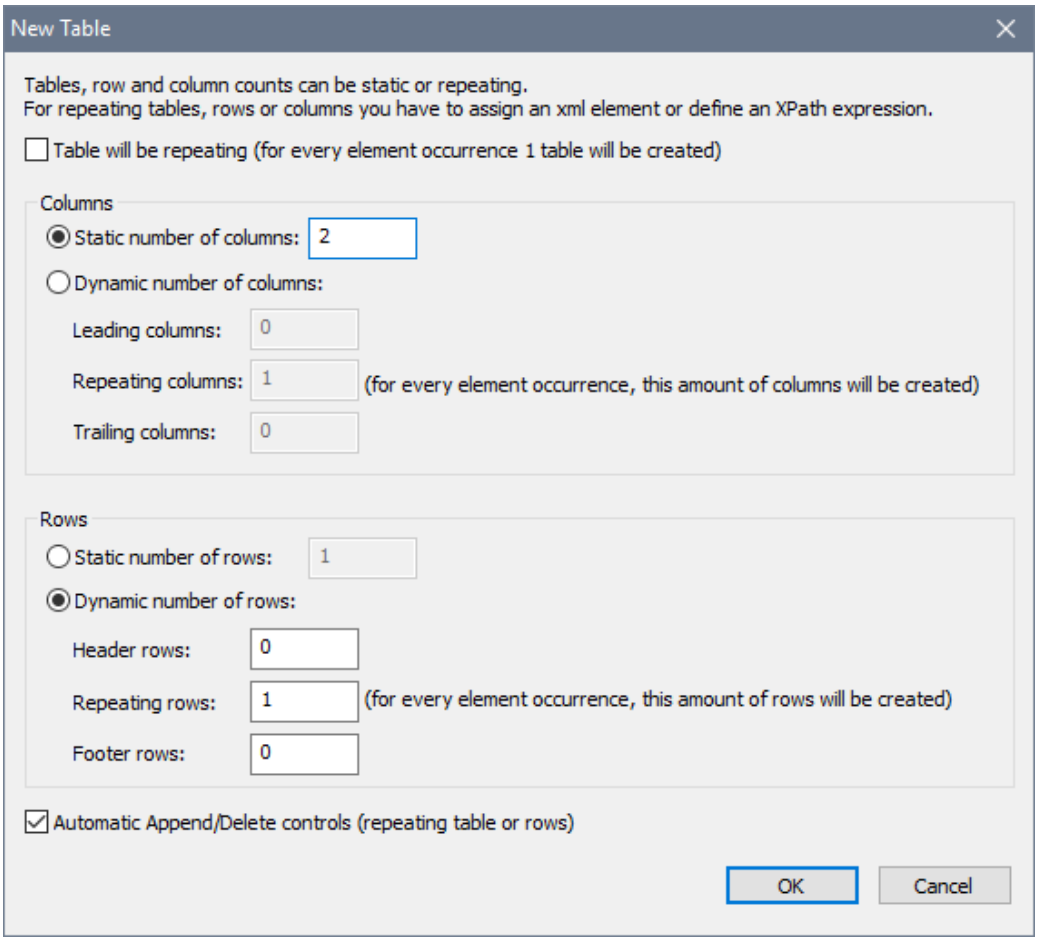
- A *table with dynamic rows* can have a header and/or footer that applies to the entire table. It is a header/footer for the table. If a header/footer is required for a *repeating table*, they can be manually added outside the repeating table. If added inside the *repeating table*, then these will repeat with each table for each element occurrence.
- Since tables are typically rendered with space above and below them in device displays, *repeating tables* will contain vertical space between its each pair of repeated tables.

To convert a *repeating table* to a *table with dynamic rows*, 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**.

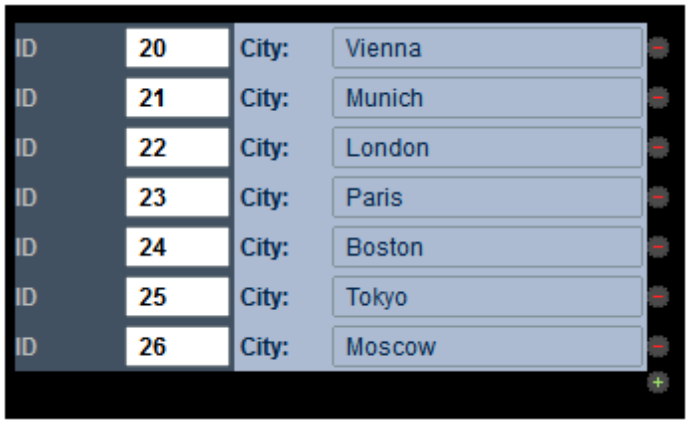
Creating a table with dynamic rows

Set up a table with dynamic 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 a dynamic table row group..



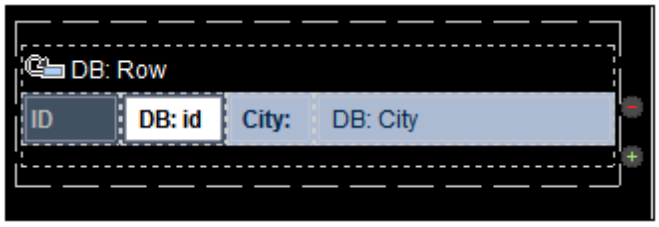
2. Specify the number of columns that the table will have and the number of rows that the row group will have (the rows together constitute the row group). You can also specify that the table should have a header and/or footer.
3. Specify whether automatic Append/Delete controls are to be added. If added, 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*).



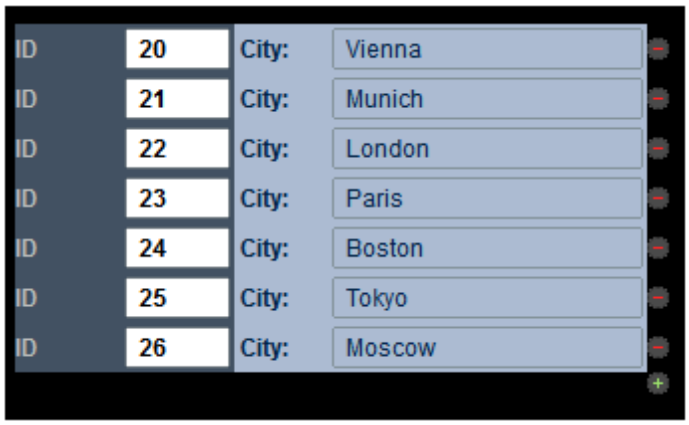
4. On clicking **OK** in the New Table dialog, the table is added to the design.
5. The row group must now be associated with the repeating element from the data source.

Associate the row group with a repeating element by dragging and dropping the element from the Page Sources Pane into the table. Each occurrence of this element will generate a row (group) in the table. The current occurrence will also be the context node of XPath expressions located inside the row group.

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 dynamic row (*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 data 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.



This dynamic row produces the following structure in the MobileTogether solution.



For information about spanning dynamic columns,see the section Row/Column joining and spanning.

Table restructuring commands are available in the context menu of the table. Table formatting properties are available in the Styles & Properties Pane. See also Repeating Tables and Dynamic Columns.

Dynamic 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 data source. When the table is rendered, the number of columns will correspond dynamically to the number of occurrences 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 table containing dynamic rows*). 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 the column is static 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 occurrences in the data 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 data 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.

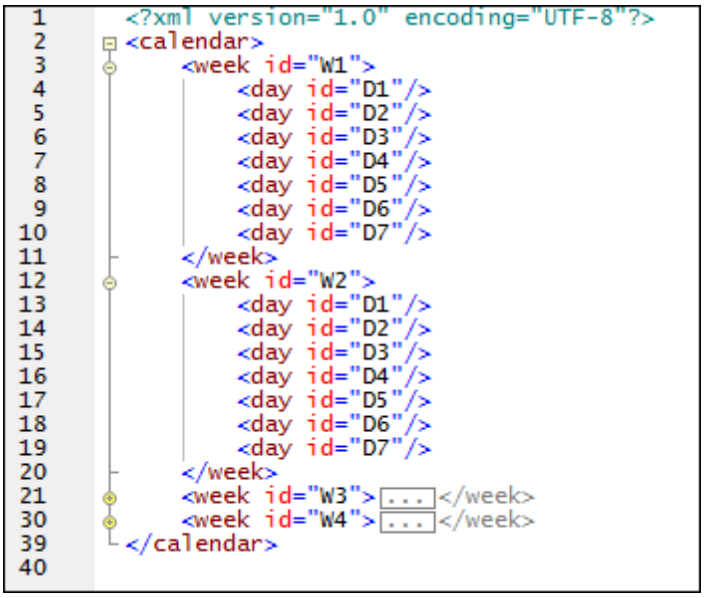
	<day>	**<day>**	**<day>**
<week>			
<week>			
<week>			
<week>			

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 data source for use elsewhere in the document.

Example: dynamic columns for days inside dynamic rows for weeks

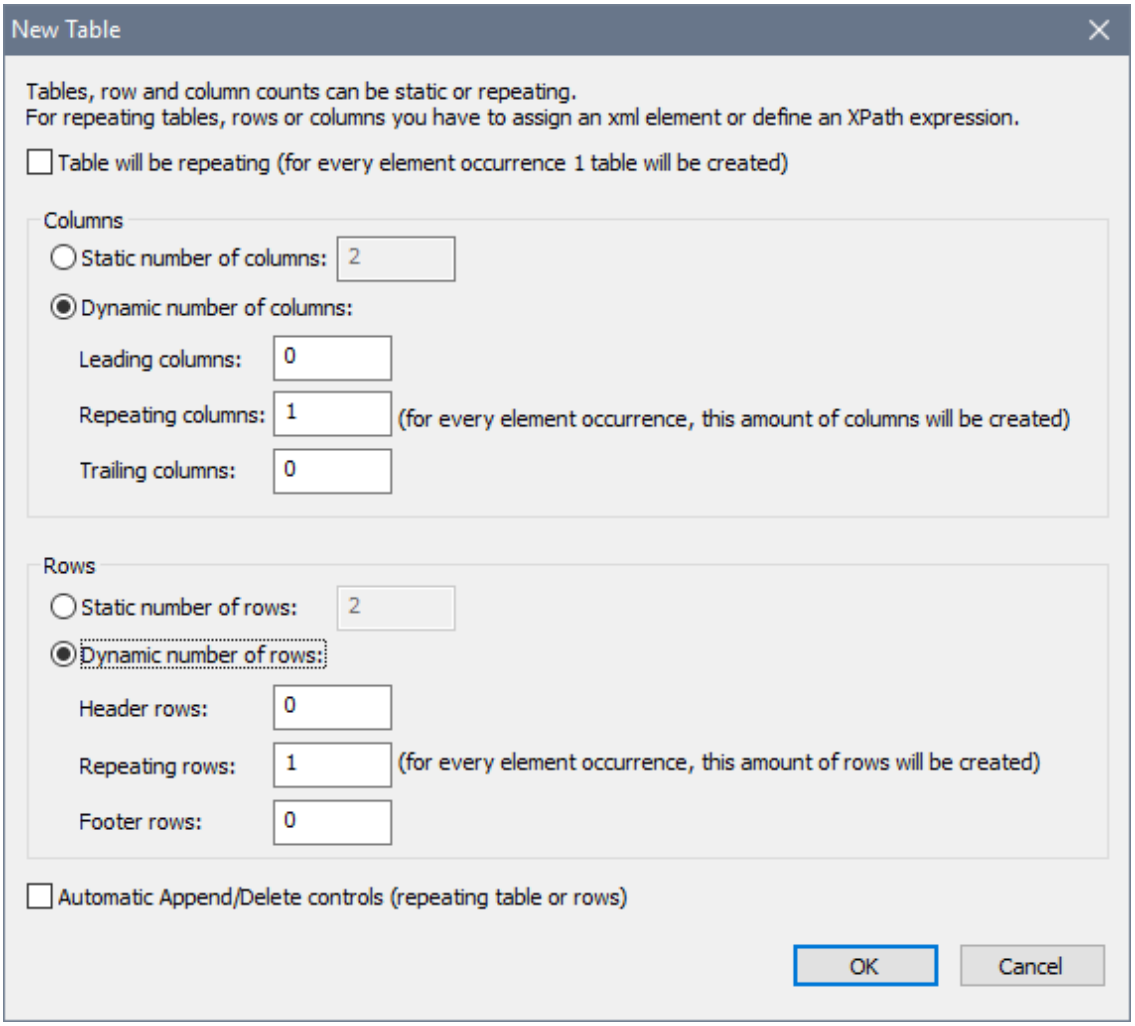
The screenshot below shows a **calendar** element that contains four **week** elements, with each **week** element containing seven **day** elements. We can make a table containing dynamic rows for **week** elements and dynamic columns (inside each **week** element) for the **day** elements. Notice

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.



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 occurrence 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.

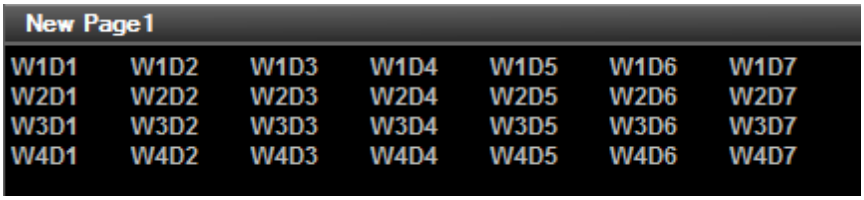


The table will be created in the design. The screenshot below shows the completed design of a table with dynamic rows and columns. Note the following points:

- 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 data source nodes that will provide the data for the table's rows and columns. The element associated with the column must, in the data 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.

- Also note the XPath expressions that are used to associate the row and column fields with data 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. 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:

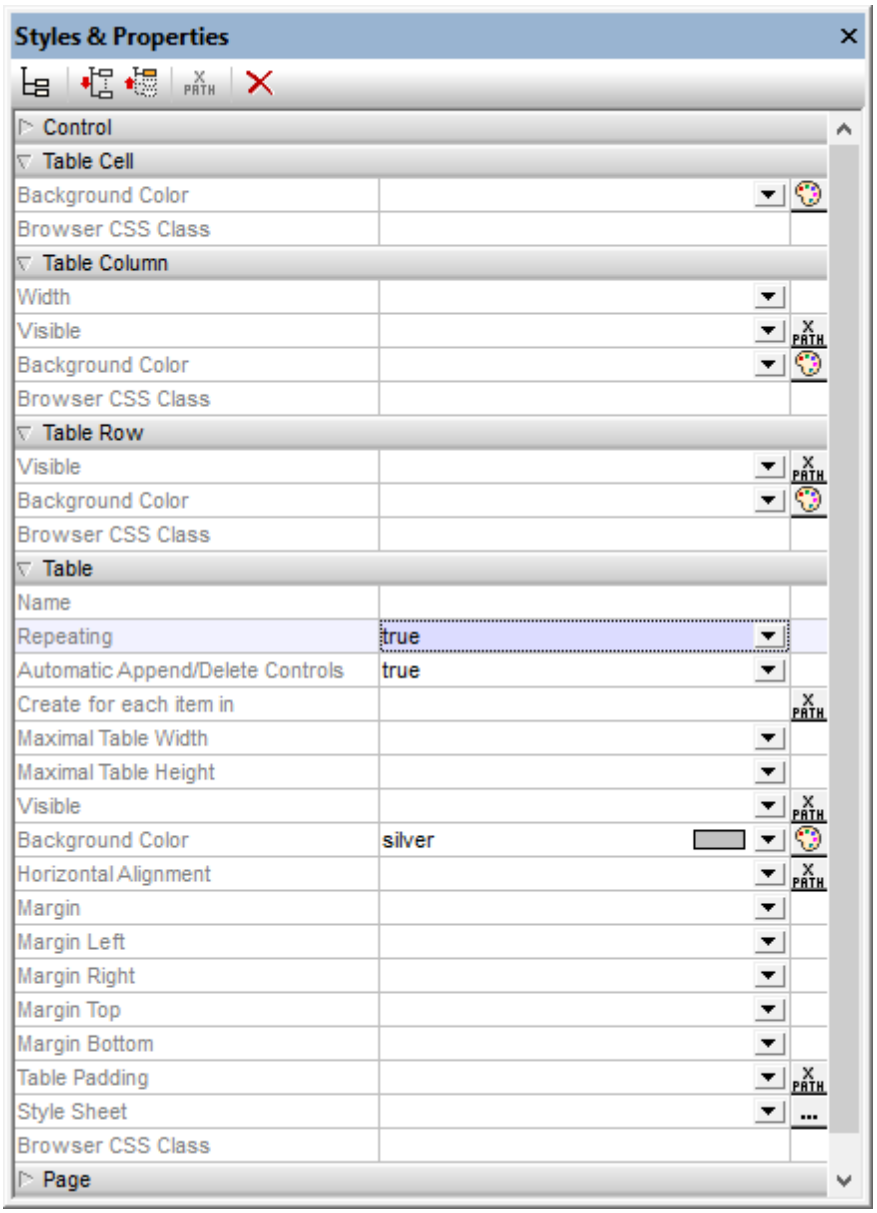


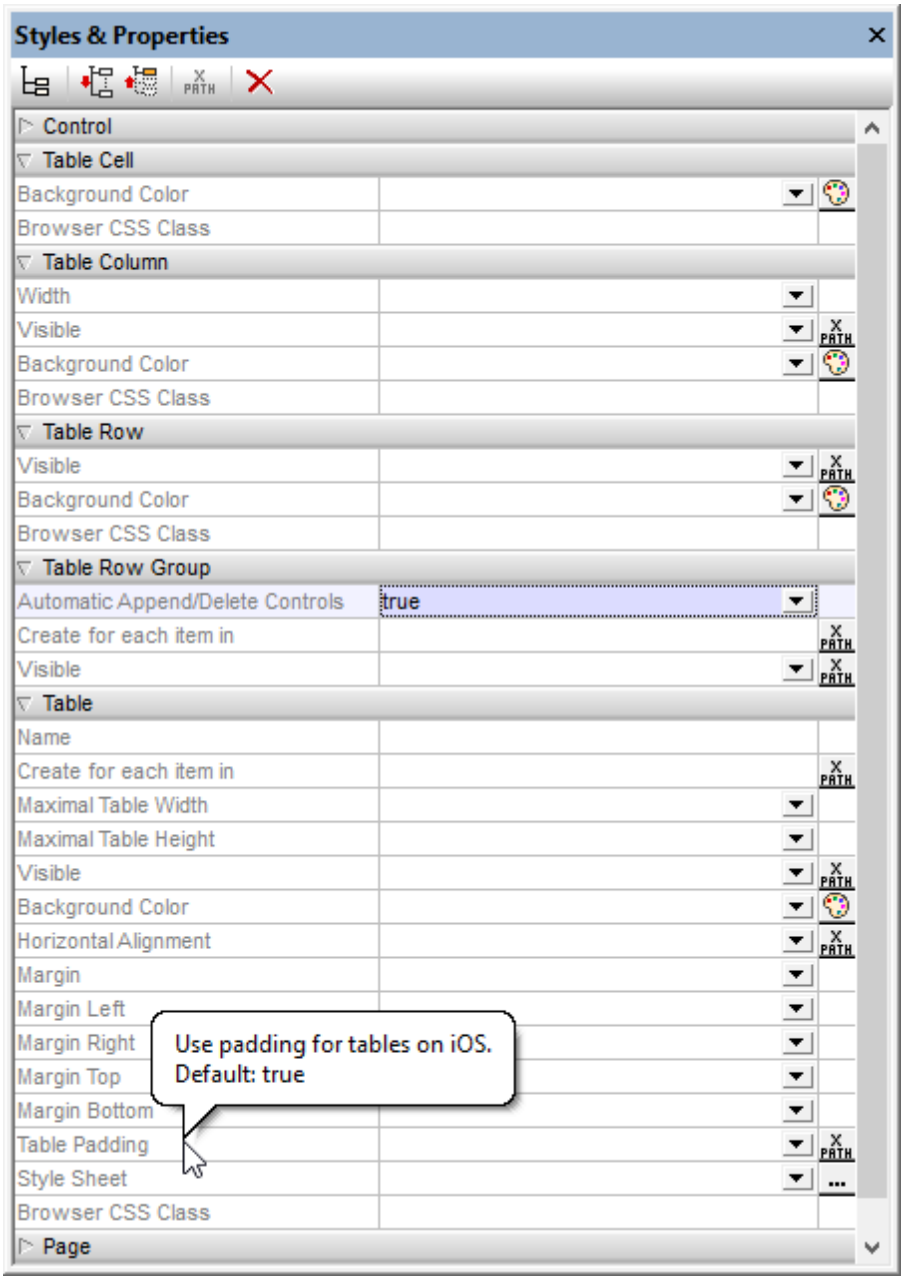
- Each **week** element in the data source is displayed in a row (**W1** to **W4**).
- Each **day** element within a week is displayed within the appropriate column (**D1** to **D7**).
- The concatenation of the two IDs is executed separately for the 28 cells from **W1D1** to **W4D7**.

For information about spanning dynamic columns,see the section Row/Column joining and spanning.

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 properties of static tables and repeating tables; the screenshot below right shows the properties of dynamic tables. Dynamic tables have an additional set of *Table Row Group* properties. All these properties are described in the section Controls | Table.





Property: Repeating

The **Repeating** property is available for repeating and static tables, and defines whether the table is repeating or static; it is not available for dynamic tables. The property's values are `true` or `false`.

The value of this property is automatically assigned at the time a repeating or static table is created. A repeating table has a **Repeating** property of `true`, while a static table 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 (repeating tables) and for table row groups (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 data source tree. This is an alternative to associating a repeating table (or table row group) with a data source node—an association made by dragging and dropping the node onto the table. 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 data 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 (**Project | Simulate Workflow**), and, in the Simulator dialog that appears,click **Evaluate XPath** and then **Evaluator**.

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` property (available in the Styles & Properties Pane) 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] *Column-1*	**Node[1]** *Column-2*	**Node[2]** *Column-1*	**Node[2]** *Column-2*	...	**Node[n]** *Column-1*	**Node[n]** *Column-2*

If the dynamic column group were spanned, by setting the `Spans Column Groups` 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** command 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 associated with the **week** element (which, in the data source, is the parent of the **day** element). Since the **week** element repeats, a

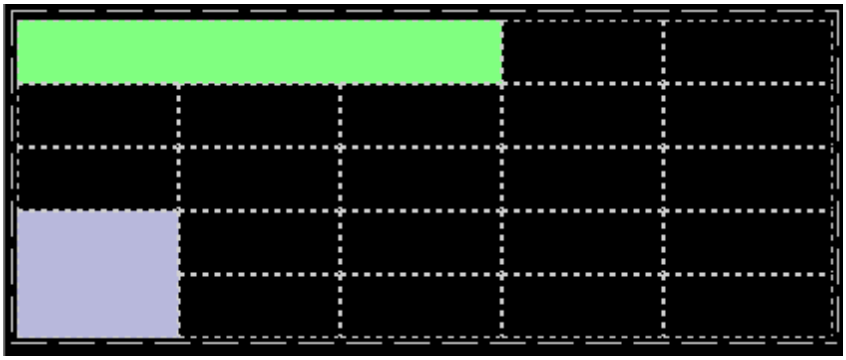
new table will be created for each **week** element. If, in the data source, there are multiple **day** child elements of the **week** element, and if, in the design, the dynamic columns of the column group are not spanned, then the table (for each **week**) generated from this design will have as many columns as there are **day** child elements. If, however, you set the `Spans Column Groups` 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.

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.

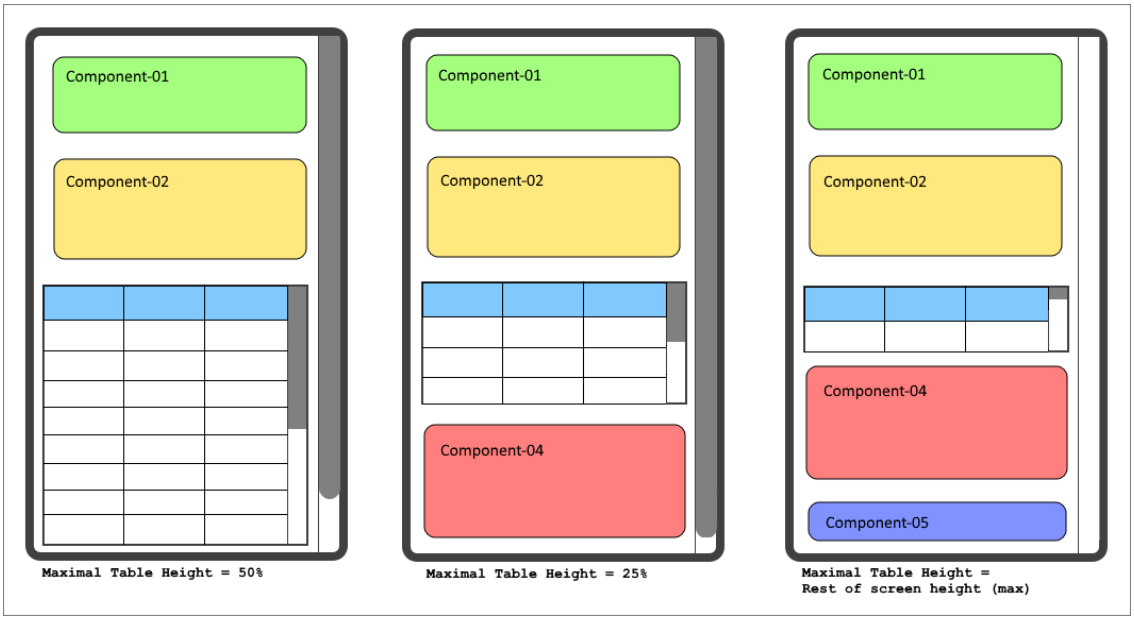
Vertical scrolling

The **`Maximal Table Height`** 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 for an example.



The **Maximal Table Height** 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** property becomes available after a value has been set for the **Maximal Table Height** property **and** no **Maximal Table Width** value is set. The **Scroll Vertically** 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 **Maximal Table Height** 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.

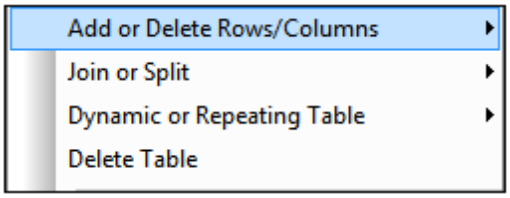
The **Row Group Chunk Size** property becomes available only if there is a repeating row-group in the table and after a value has been set for the **Maximal Table Height** property of scrollable tables. It enables you to specify the number of row groups that are loaded at a time. When the user scrolls 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 **`Maximal Table Width`** 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 Context Menu

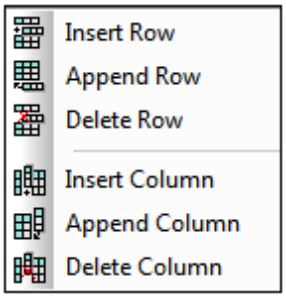
The context menu of all tables (static, repeating, and tables with dynamic rows and/or dynamic columns) has the same commands (*see screenshot below*). These commands allow the structure of the table to be modified after the table has been created.



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.

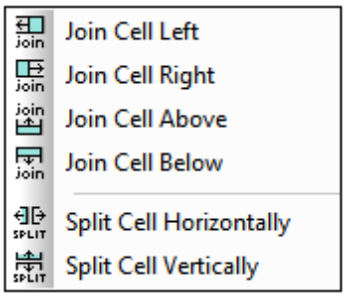
The currently selected row/column can also be deleted.



These commands are available when a row or column of any kind of table (static, repeating, or tables with dynamic rows and/or dynamic columns) is selected.

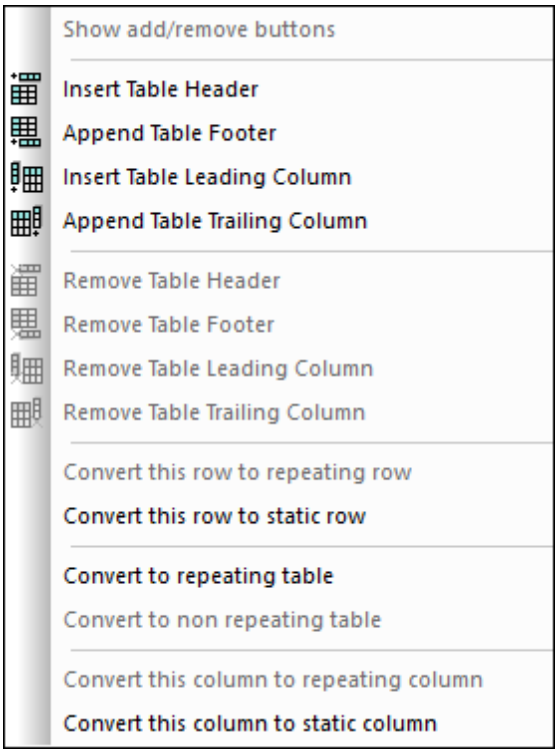
Join or Split

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, repeating, and tables with dynamic rows and/or dynamic columns).



Dynamic or Repeating Table

Hovering over the **Dynamic or Repeating Table** command displays a submenu (*screenshot below*).



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 tables and tables with dynamic rows).
- Insert/remove a header/footer (in tables with dynamic rows), and insert/remove a leading/ trailing column (in tables with dynamic columns).
- If the selected row ***is not*** a dynamic repeating row, convert it to a repeating row.
- 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 and static. (Note that the rows and columns of a static table can be converted, respectively, to repeating rows and repeating columns).

Delete Table

Deletes the selected table.

11.2 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, 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 is the property of the Image control 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.
- Base64-Encoded Images describes how to use Base64-encoded images in your design.
- Exchangeable Image File Format (Exif) 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 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 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 lists the ways in which images can be stored in databases.

These mechanisms are enabled by powerful Image actions and image-related Altova XPath extension functions.

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 into the design.
2. In the Styles & Properties Pane, set the image property **`Image Source Type`** 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`** property. If an image file is being referenced, specify the URL. If a Base64-encoded image is being referenced, use the **`Image Source`** 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`** property selected, click the XPath toolbar button of the Page Sources Pane, 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 onto the Image control.

Note: Every time an image source is changed (for example, by a user selection), a Reload action 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`** property. For an example of inserting images via a URL, see the QuickStart Tutorial.

Embedding URL-sourced images in the design file
If an image is sourced via a URL (and not as a Base64-encoded image), the image can be embedded in the design file. Use the **`Embed Image`** property of the Image control. If this property is set to true, the image is converted to Base64-encoding and embedded in the design file.

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** 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 onto the Image control.

See the next section, Base64-Encoded Images, for an example of how to use Base64-encoded images.

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>
```

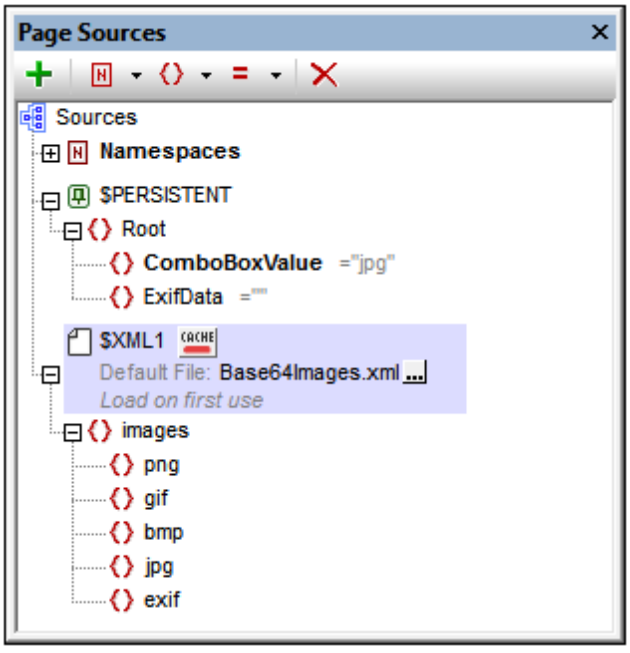
To insert a Base64-encoded image, the XPath expression of the Image control's **Image Source** property must resolve to the image's Base64-encoded text string. You can also drop an XML node that contains the image's Base64-encoded string from the Page Sources Pane onto the Image control. The example 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 *(My) Documents* 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.



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.

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` 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 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 is created with a simple list of values.
- The **`Image Source Type`** property of the Image control is set to `base64`.
- The **`Image Source`** 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`** 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. 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`**. 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`** 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), describes the remaining part of the design, which deals with Exif data.

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 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`** 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`** function for details. To find just the dimensions of images, use the MobileTogether XPath extension function **`mt-image-width-and-height`**.)
- An Altova XPath extension function called **`suggested-image-file-extension`** 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 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 will be lost if the image is resized or rotated.

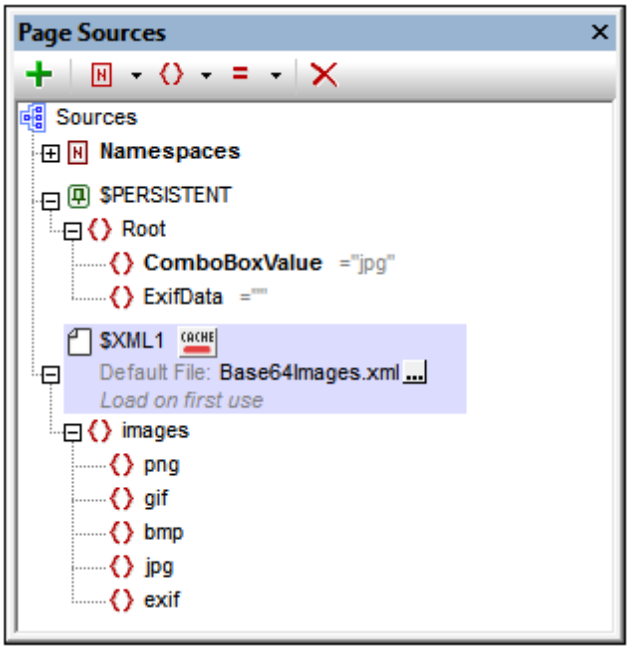
Example file: Base64Images.mtd

The design file **`Base64Images.mtd`** is located in your *(My) Documents* 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

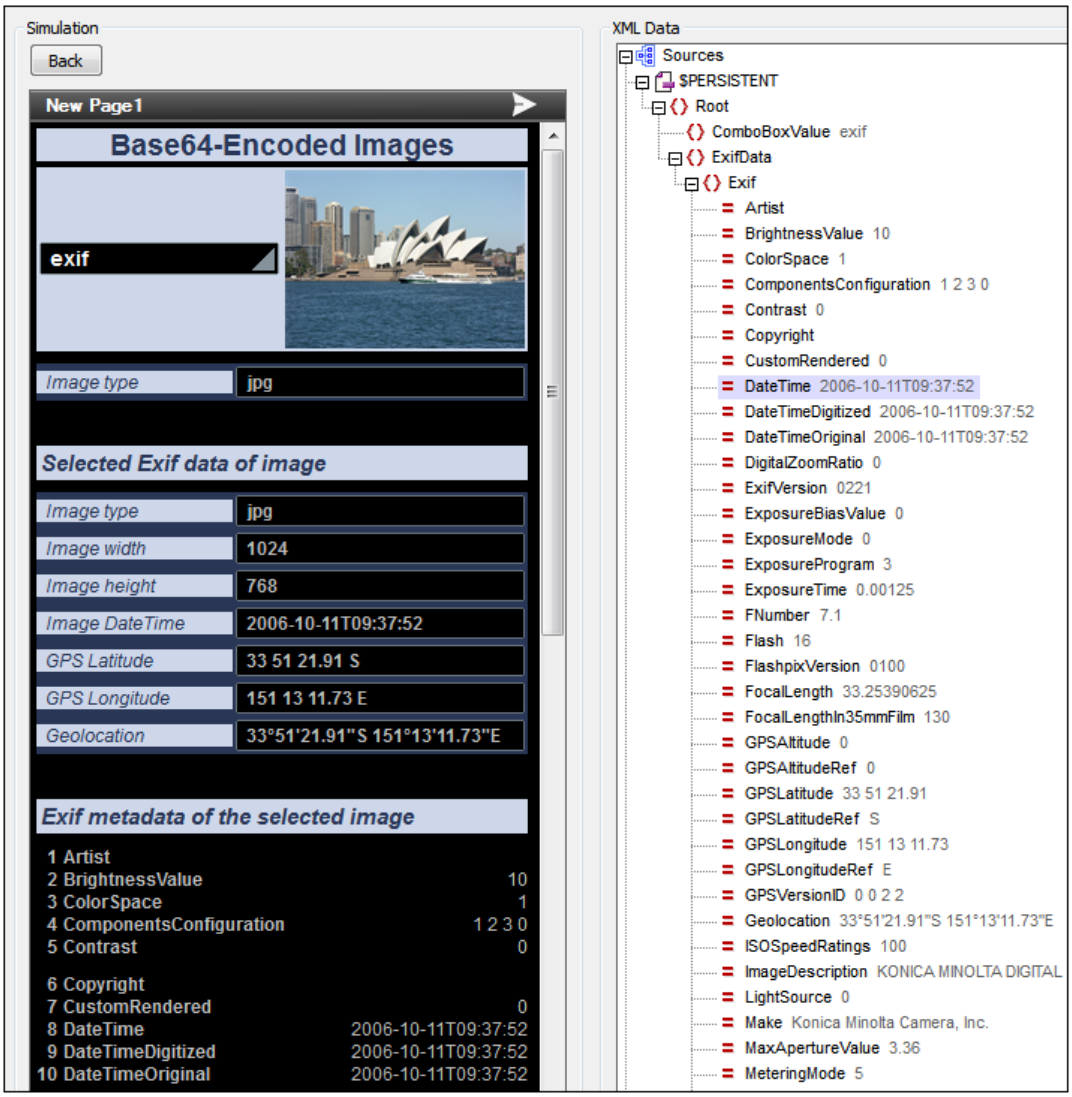
The design file uses Base64-encoded images that are stored in the XML file

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).



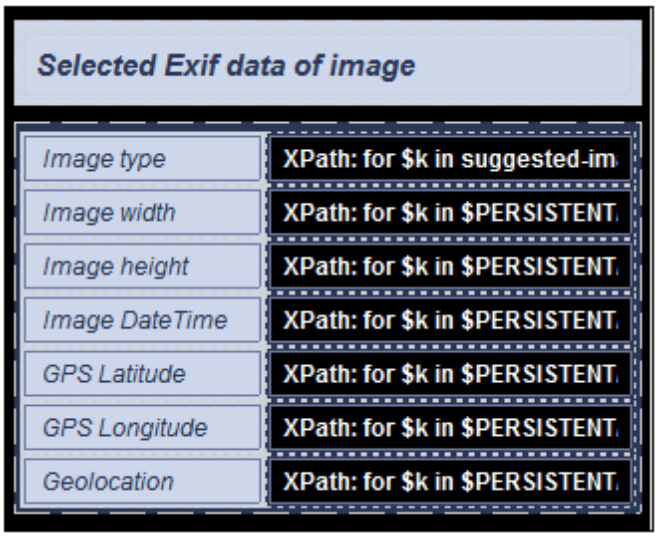
The top part of the design (*screenshot below*) has a label for the page title, and two tables. The design of this part of the page is described in the previous section, Base64-Encoded Images. The objective is to allow the end user to select an image type in the combo box. This selection determines which 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, for a detailed description of other parts of the design.



- *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.



- The *Image type* information is obtained from the Base64-encoded text string by using the Altova XPath extension function **suggested-image-file-extension**. 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"
```

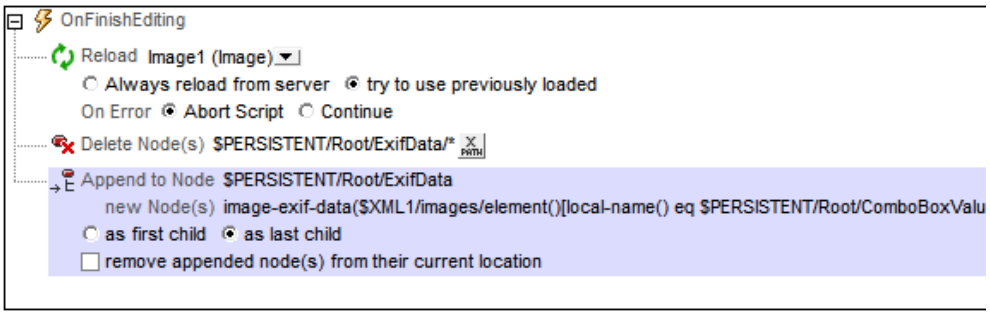
  The expression provides alternative return strings depending on whether the function returns a non-empty string or an empty string. If a non-empty string is returned, then the string is displayed; if an empty string is returned, an appropriate message is displayed.
- All the other XPath expressions in the table (besides the first row) use the Altova XPath extension function **image-exif-data** 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** 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** function, see its description in the Altova extension functions section.
- Note the last *Geolocation* value in the table. It is obtained via an Altova-created

`Exif/@Geolocation` attribute.

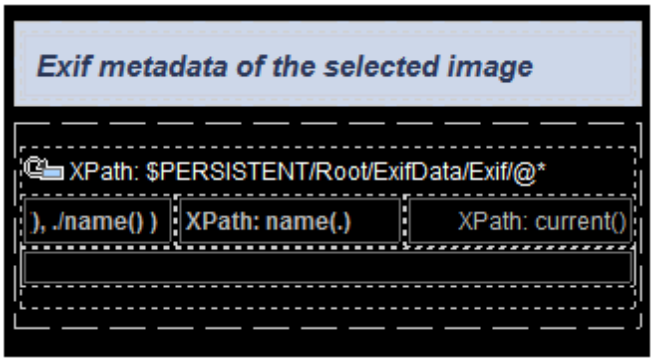
- The `$PERSISTENT/Root/ExifData` node is populated with the Exif data by appending a child node to it that contains the result of the **`image-exif-data`** function. This is done by specifying an Append Node action on the combo box that selects which image to display (*see screenshot below*). The action is triggered by the `OnFinishEditing` event of the combo box.

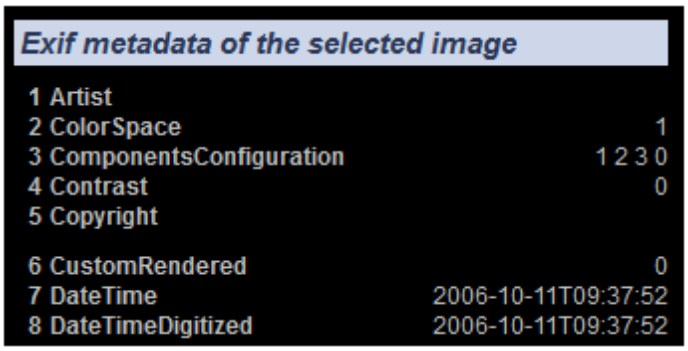


Note the following points:
(i) The XPath locator expression in the Append Node action locates the node in the XML file that has the same name as the string in `$PERSISTENT/Root/ComboBoxValue`.
(ii) The `$PERSISTENT/Root/ExifData` node is deleted before the Exif data (in the `Exif` node) is appended to `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`** 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`** function: **`$PERSISTENT/Root/ExifData/Exif`**`/@*`.

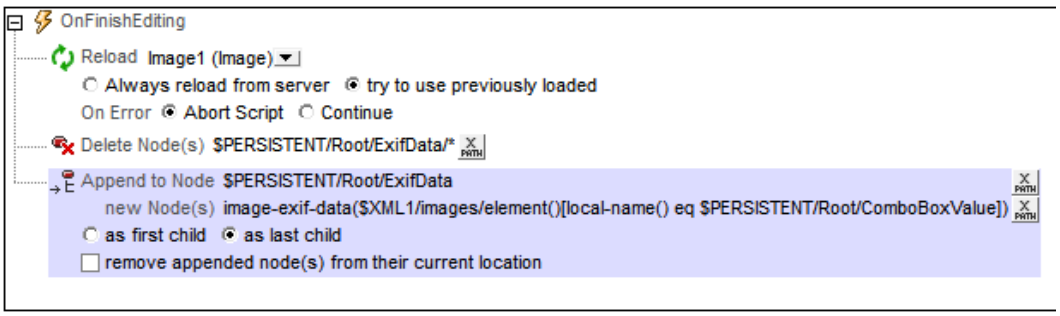




- The table's first column contains the index position of the current attribute: `index-of(../@*/name(), ./name() )`
- The second column contains the name of the current attribute: `name(.)`
- The third column contains the value of the current attribute: `current()`
- Not all images contain the same 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 to be retrieved.
- If we know the names of the attributes that are returned, we can access its value by using the **`image-exif-data`** 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`** 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`** function.
- This is done by specifying an Append Node action on the combo box that selects which image to display (*see screenshot below*). The Append Node action is triggered by the `OnFinishEditing` event of the combo box.



- The XPath locator expression in the Append Node action 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`** 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`** function like this: **`image-exif-data(Base64String)/@WantedAttribute`**. Note that the function returns the **`Exif`** element.

Images Chosen by End User

The Let User Choose Image action 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 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, 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).

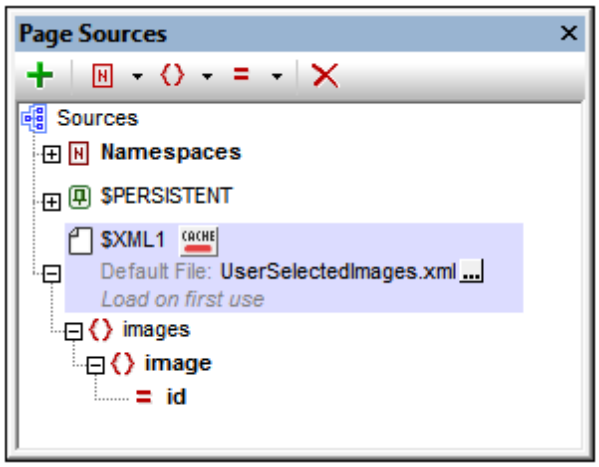
Note: If an image is displayed in the design, then, every time the image source is changed (for example, by a user selection), a Reload action for the image is required in order to display the new image in the design.

Example file: UserSelectedImages.mtd

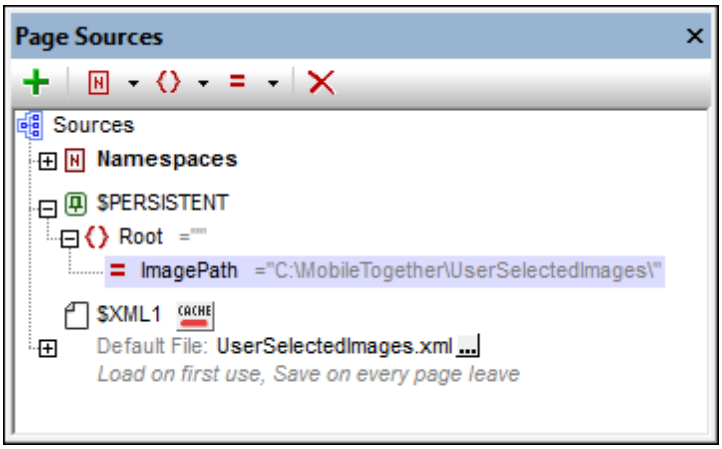
The design file **UserSelectedmages.mtd** is located in your *(My) Documents* MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\Images**. You can open this file in MobileTogether Designer, run it in the simulator (**F5**), and look through the design definitions. You will also need to 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 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 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**.

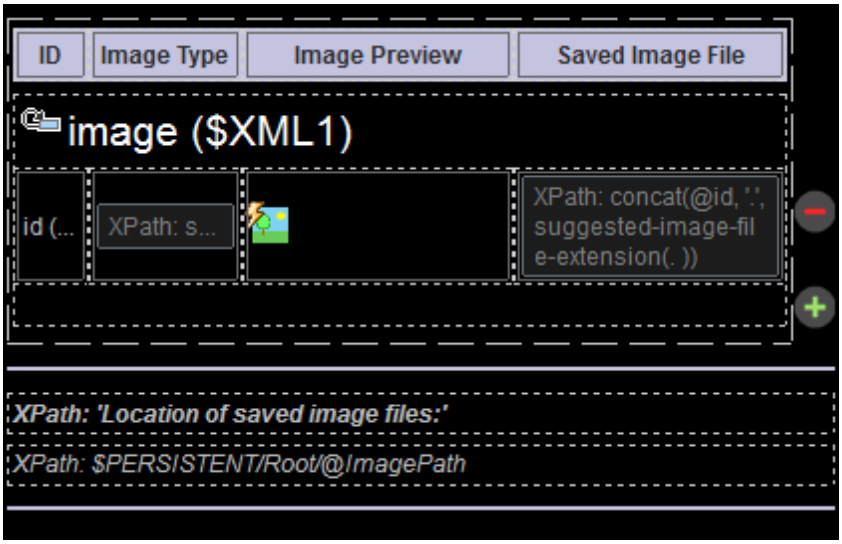
The design has a single data source, an XML file called **UserSelectedImages.xml** (also located in the `Tutorials\Images` folder). The structure of the XML document is shown in the screenshot below. The `images` element can have multiple **image** child elements. Each `image` element has an `id` attribute, 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.

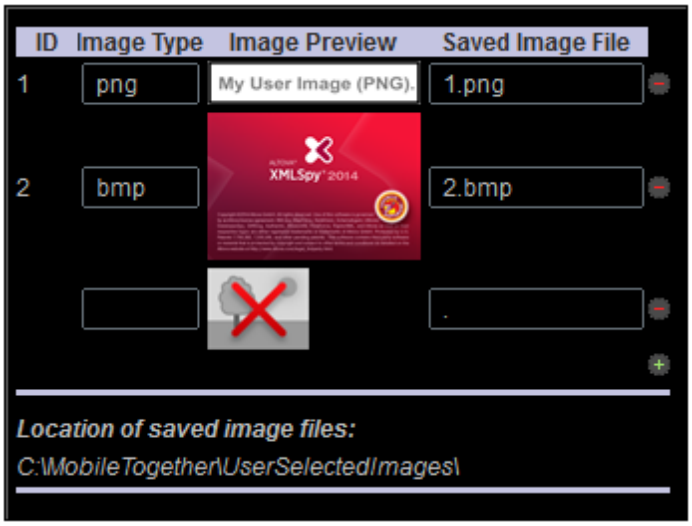


A node, **$PERSISTENT/Root/@ImagePath**, has been created to hold the path of the folder where images will be saved (*see screenshot below*). It has been set to a default value of **C:\MobileTogether\UserSelectedImages**. You can modify this path directly in the Page Sources pane of the design if you wish to change the location of the folder where images are saved: double-click in the name and edit it.



The design (*screenshot below left*) consists of a label that displays the page title and a dynamic table. 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 it own row. The screenshot below right shows the solution being run in a simulator. A description of the design is given below.





Note the following points about the design:

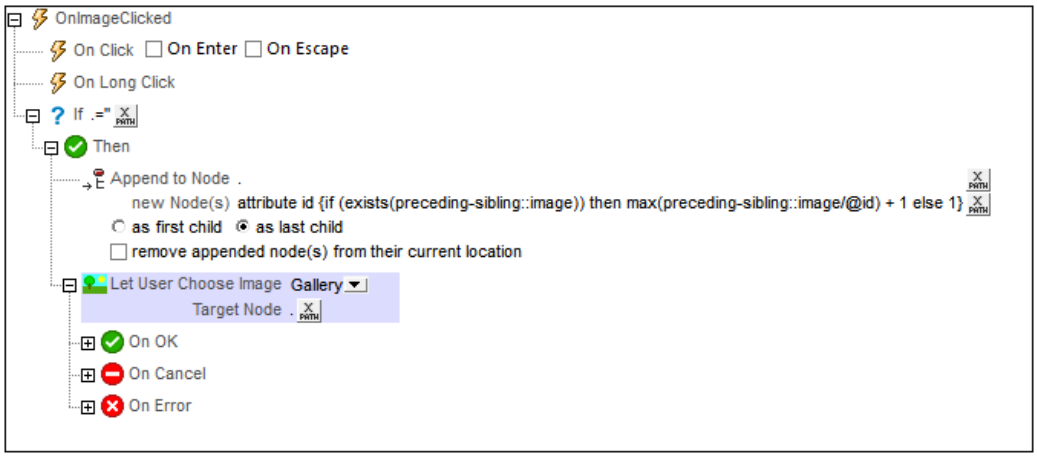
- The dynamic table has **Add/Remove** buttons (added via the table's context menu). This enables the user to add a new `image`-row and to delete any `image`-row.
- Each `image`-row has four columns: *ID*; *Image Type*; *Image Preview*; and (the name of the) *Saved Image File*.
- When a new `image`-row 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 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 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**. 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. 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 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** to provide the image file extension.

Actions of the OnImageClicked event

- The `OnImageClicked` event of the Image control has been assigned the actions shown in the screenshot below.



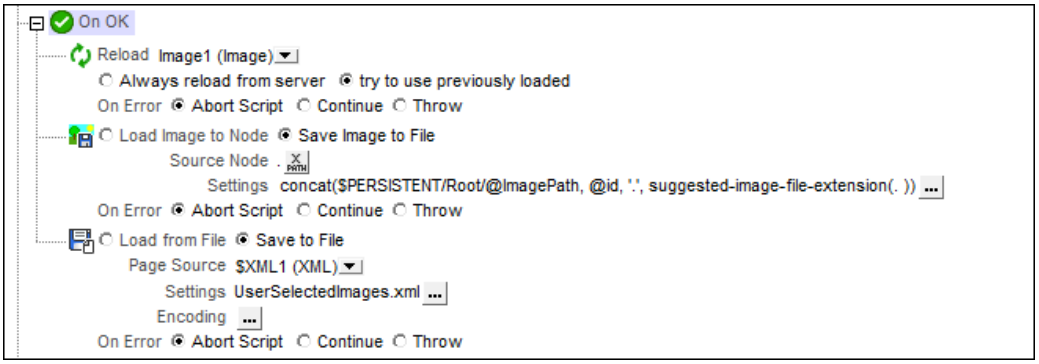
- The `If` condition specifies that if the current node (the `image` element) is empty, then, when the image is clicked, a new **id** attribute is created and appended as a child to the currently empty `image` element. (The empty `image` element was added when the user added a table row (*see the Simulator 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 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 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 for the image; (ii) a Load/Save Image to File action that saves the image from the data source node to an image file; (iii) a Load/Save to File action which saves the data in the source tree (on client/server) to the data source file.

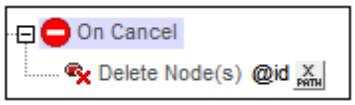


- A Reload action is set for the Image control. This will cause the image specified in the `Image Source` property of the Image control to be reloaded. Since the value of the `Image Source` property is set to the current node, and since the current node is the `Image` element which is the target node 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 (*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.

- 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`** provides the file extension.
- The Load/Save to File action 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. 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 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*).

Transforming Images

A Base64-encoded image can be transformed (resized, rotated, and modified for quality/filesize) with the Altova XPath extension function **mt-transform-image**:

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.

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** 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, specifying that the child actions be performed on the server. All the child actions of this Execute On action will then be carried out on the server. You can use an action such as Update Node 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.

Images in Databases

Images in databases can be stored in Base64 format, which is a text format into which binary data can be encoded.

11.3 Audio, Video

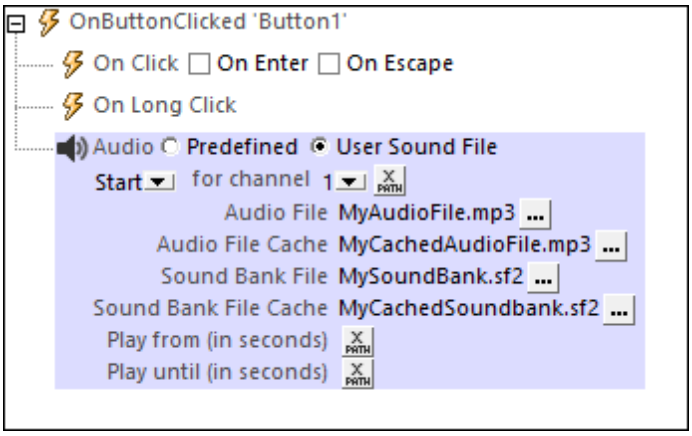
This section provides an overview of MobileTogether's audio and video features. It is organized into the following sections:

- Audio Playback
- Audio Recording
- Video Playback
- Audio/Video Formats

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. 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 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.



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, 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. 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 action to load the binary audio/video data to a data source node; (ii) use the Save Binary action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 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` project property. Since each of these events apply across the entire project, each event could be triggered by the audio on any channel. The `$MT_AudioChannel` dynamic variable contains the number of the channel that triggered the event. So, for example, if the user starts playback of an audio file that runs on Channel 2, you can use the `$MT_AudioChannel` 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` 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).
- `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`. 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*) or paused.

Audio-playback-related MobileTogether XPath extension functions

The following audio-playback-related MobileTogether XPath extension functions are available:

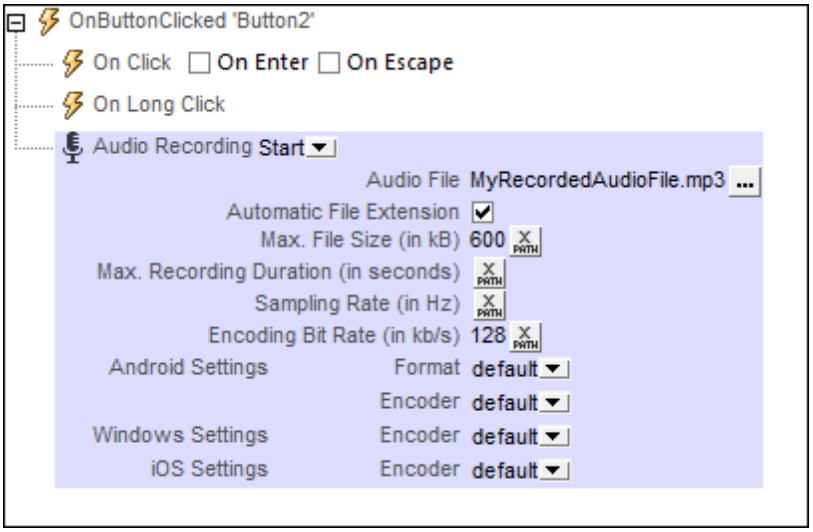
- **mt-audio-get-current-position(** ***ChannelNumber*** as `xs:integer` **)** as `xs:decimal`
- **mt-audio-get-duration(** ***ChannelNumber*** as `xs:integer` **)** as `xs:decimal`
- **mt-audio-is-playing(** ***ChannelNumber*** as `xs:integer` **)** as `xs:boolean`

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.

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 (*see screenshot below*) and stopped via the Audio Recording (Stop) action. See the description of the Audio Recording action 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.



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, 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, 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`**. 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

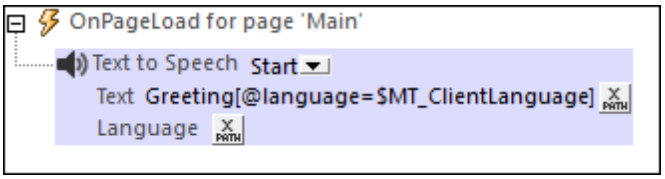
The following audio-recording-related MobileTogether XPath extension function is available:

- **`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.

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 (*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.



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 enable actions to be set on the following events: `OnTextToSpeechStarted`, `OnTextToSpeechError`, `OnTextToSpeechCompleted`. These events enables 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 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.

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 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 for details.
2. Video actions 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 on a page. Each Video control is identified by a name, and is assigned a video source via a URL. The name of a Video control will be used in the Video action to indicate on which Video control the action is to be performed.

The following Video control 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 (for example, on a Button).
- **`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. 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.

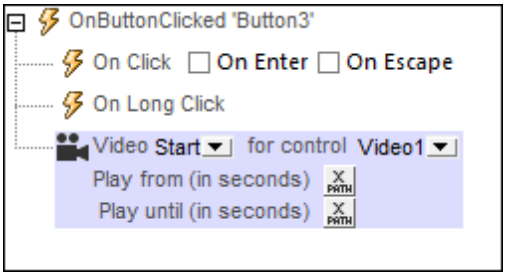
Note: Multi-channel audio/video playback is not supported on Windows Phone. Only one audio **or** video file can be played at a time: this is the file that was started last.

Note: Audio and video files **cannot** be deployed to MobileTogether Server via the MobileTogether Designer project's Deploy to Server mechanism. 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 action to load the binary audio/video data to a data source node; (ii) use the Save Binary action to save the data in this node to a file on the client device; (iii) use audio/video playback actions 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 (*screenshot below*): (i) identifies the Video control 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 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.



Note: If a video stream is playing when a solution is suspended, then playback is paused. Playback continues when the solution is resumed.

Video playback events

Video playback events are defined on each Video control and apply to that Video control. 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) 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`**. 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*) or paused.

Video-related MobileTogether XPath extension functions

The following video-related MobileTogether XPath extension function is 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.

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)**

11.4 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—contains information that other (active) NFC devices, such as smartphones, can read. Active devices can read information and send it. 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.

Android Beam™ and NFC availability on Android

Android Beam™ (hereafter Android Beam) is an app that is available on Android devices since Android 4.0. It can be used to share data between two Beam-enabled devices. Typically, the currently open content (web page, photo, contact information, etc) on one device is beamed to the other device by touching the devices back-to-back.

Note: Although Android Beam is available in the OS, it will not be available for use on a device if the device does not support NFC (for example, if the device does not contain an NFC chip).

In order to use Android Beam on a device, make sure that (i) the device supports NFC, and (ii) both NFC and Android Beam are enabled on the device. To check the availability of NFC and Android Beam, and to enable them, go to the **Settings | Network** tab of the Android device. If the *NFC* and *Android Beam* options are available, then enable them if you want to use them. If both options are not available and if the OS is later than Android 4.0, then NFC is not available on the device.

Note: If both an Android device and a Windows device are NFC-enabled, then they can communicate with each other via NFC.

NFC availability on Windows and iOS

- *Windows*: To check the availability of NFC on a Windows device and enable it, go to: **Settings | Tap+Send**.
- *iOS*: NFC is used only with Apple Pay. It cannot be used for sharing other kinds of data.

Note: If both an Android device and a Windows device are NFC-enabled, then they can communicate with each other via NFC.

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 NFC Tags Explained.

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 one device to another (from Android and Windows

devices)
- Files can be beamed from one Android-Beam-enabled Android device to another Android-Beam-enabled Android device

Note: NFC support is not available on iOS devices.

In this section

- Discovering and Reading NFC Tags
- Pushing Data to Other Devices
- NFC-Related Events
- Design Components for NFC

Discovering and Reading NFC Tags

After NFC has been started via the NFC Start action, NFC tag discovery is automatically started. If an NFC tag is discovered, then the NFC message in the tag will be automatically received and the information in the message will be stored in the **$MT_NFC** tree. This tree was created in the design when the NFC Start/Stop action was added to the design. The complete structure of the tree is given below. Note that an NDEF message can contain multiple NDEF records, and the NDEF records can be recursive. If the NFC tag information that is received does not contain information to fill all the attributes of the **NDEFMessage** or **NDEFRecord** elements, then these attributes will not be created in the **$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** 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.

Pushing Data to Other Devices

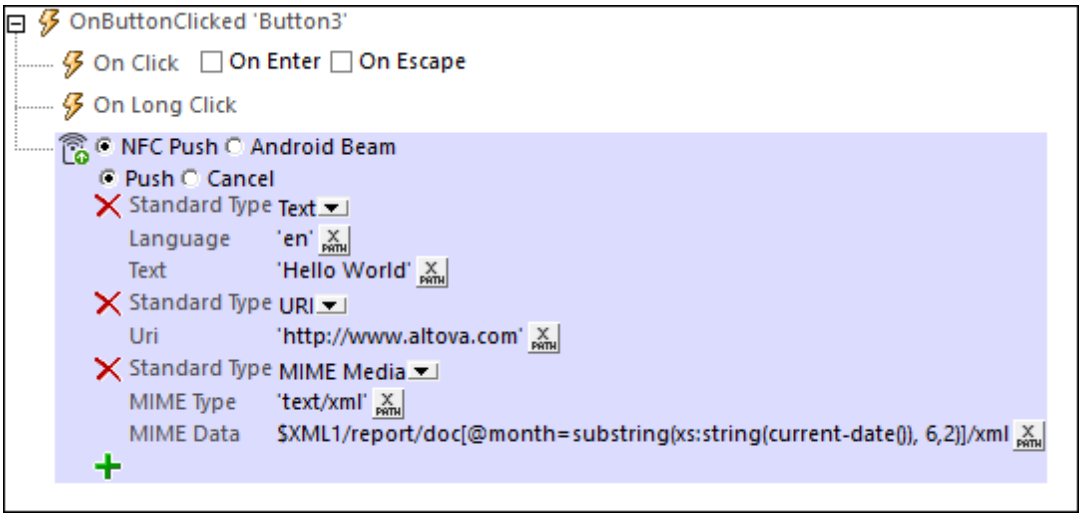
A MobileTogether solution can be used to transmit data from the device running the solution to another NFC-enabled device. The steps of the end-user procedure are the following:

1. On the sending device, start the solution.
2. Trigger the NFC Start action (for example, by tapping a button).
3. Place the back of the sending device against the back of the receiving NFC-enabled device.
4. Trigger the NFC Push action to start the transmission (for example, by tapping a button). Once an NFC Push action has been started, the NFC message or file can be sent multiple times (that is, to different devices). All that the end user needs to do is to place the receiving device within NFC range of the sending device. This continuous sending is stopped when NFC is stopped, or when an NFC Push is canceled (which is done by adding a new NFC Push action with the *Cancel* option selected; *see screenshot below*).

How it works

NFC data-transmission is defined in the NFC Push action (*see screenshot*), and can be done in the following ways.

- A message of a specifically defined type can be sent via the *NFC Push* option (*see screenshot below*). Available for Android and Windows devices. See the description of the NFC Push action for more information about these message types.
- A file can be beamed from one Android-Beam-enabled device to another Android-Beam-enabled device. Available for Android devices only.



See the NFC Push action for details of how to define the action's settings.

Note: NFC data-transmission is not supported on iOS device.

NFC-Related Events

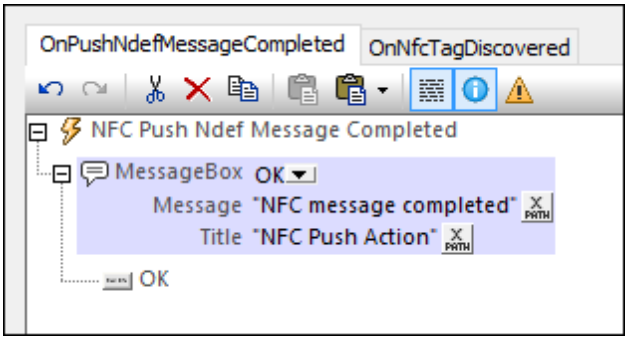
Two NFC-related events (*screenshots below*) provide valuable functionality:

- **OnPushNdefMessageCompleted** specifies what action/s to carry out when the transmission of NFC data (via NFC Push) has been completed.
- **OnNfcTagDiscovered** specifies what (additional) action/s to carry out when an NFC tag is discovered.

To access these actions, go to Styles & Properties Pane | Project Properties | 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. It can be used (i) to specify subsequent action to take, or (ii) to inform the user about the completion of the transmission. The screenshot below shows how the event can be used.

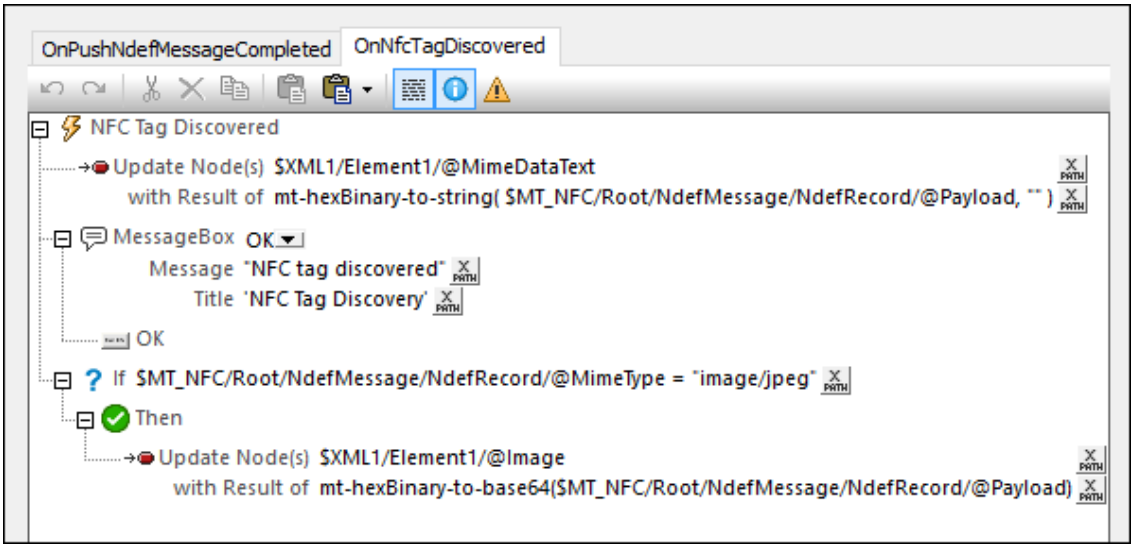


See also Pushing Data to Other Devices.

OnNfcTagDiscovered

This event is triggered when an NFC tag is discovered. When a tag is discovered, the information in it is automatically read and stored in the **$MT_NFC** tree. This event enables you to specify additional actions to carry out. For example, as shown in the screenshot below, source-tree nodes can be updated with data from the **Payload** attribute of the **$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 **mt-hexBinary-to-string** can be used to obtain the text string before placing the string in a source-tree node (*see the first Update Node 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 by using the **mt-hexBinary-to-base64** extension function.



See also Discovering and Reading NFC Tags.

Design Components for NFC

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*** 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; it is not automatically started.

  The ***NFC Stop*** 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.

- NFC Push action

  The NFC Push action enables data to be transmitted from the device running the solution to another NFC-enabled device. Additionally, if Android Beam is enabled on two Android devices, then files can be beamed from one device to the other. The NFC Push action defines the message to push or the file to beam. When the action is triggered, the specified message or file is transmitted via NFC.

  Note: NFC data-transmission is not supported on iOS device.

  For more information, see Pushing Data to Other Devices and NFC Push action.

- $MT_NFC page source tree

  The **`$MT_NFC`** tree is created as a page source in the design when an NFC Start action is defined. The tree is automatically populated when an NFC tag is discovered. The data from the NFC tag is stored in the nodes of the **`$MT_NFC`** tree. For simulations, you can use an NFC sample file to see how data from the NFC tag is stored in the **`$MT_NFC`** tree. (See the section NFC Sample Files 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) has been completed.
- **`OnNfcTagDiscovered`** specifies what (additional) action/s to carry out when an NFC tag is discovered. For example, when this event is triggered, an Update Node 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.
- **`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 to hexBinary) and converted from hexBinary to human-usable formats (text and images). For more detailed descriptions of these functions, see MobileTogether Extension Functions.

NFC sample files for simulations

For simulations, you can create an NFC sample file 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 for more information about NFC simulations.

11.5 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.

PNs can also be sent by MobileTogether AppStore Apps—which are MobileTogether solutions that have been compiled as apps that can be downloaded from app stores—to other MobileTogether AppStore Apps.

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 and used with MobileTogether controls.

Broadly, the push notification mechanism works as follows:

- A Send Push Notification 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 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 with **an external PN key** and/or to receive specific **PN topics**. In the Send Push Notification 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 data 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 provides an overview of this part of the PN mechanism.

In this section

While The Sending Solution and The Receiving Solution 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 (apps that are compiled from MobileTogether solutions), some additional steps are required; these steps are described in the Push Notifications in AppStore Apps sub-section. Simulating Push Notifications explains how to simulate PNs when the sending and receiving solutions are not one and the same solution.

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 action for this event (*screenshot below*). The various parameters of the PN are defined in the Send Push Notification action itself. The main parameters are briefly introduced below.

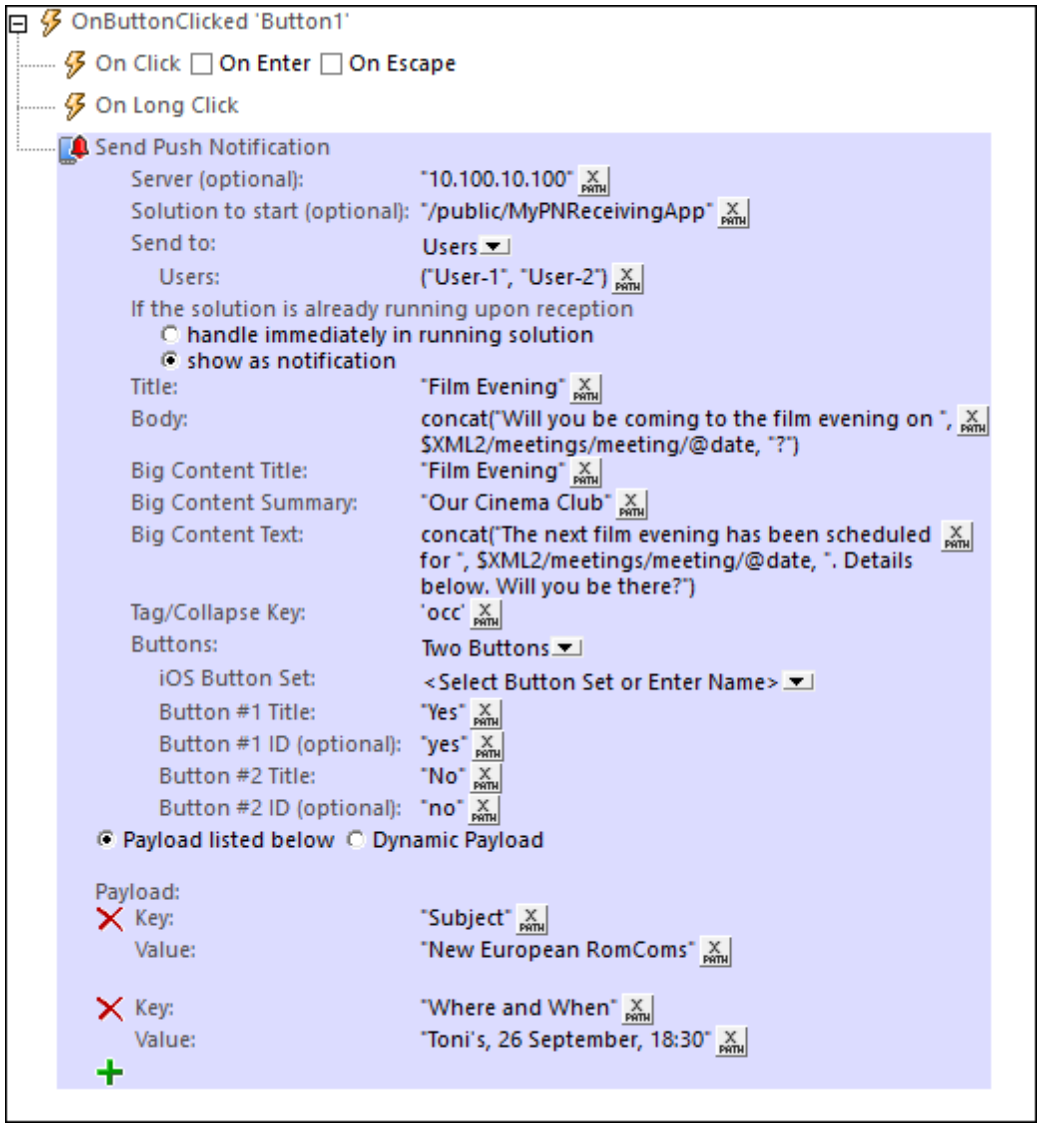
Additional to defining the parameters of the Send Push Notification action, the only other preparations required in the sending solution relate to the data to be sent in the PN. If data for the payload and/or short and big messages is added dynamically from page data sources of the sending solution, 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 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.)* The recipient list can thus be flexibly configured using any one of these selection criteria.



- 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, 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, please see Push Notifications in AppStore Apps for additional steps to take

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.

OnPushNotificationReceived event

- *At design time*, the **OnPushNotificationReceived** event is accessed via the receiving solution's project properties. 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 or If-Then-Else 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
```

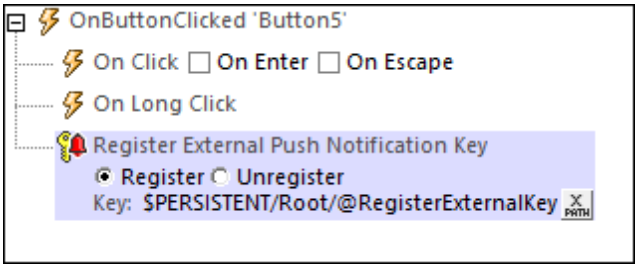
At run time:

- 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 or If-Then-Else 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 with one or more external PN keys. (Each solution on the device can register that device with a different external PN key.) This key is a text string that is generated by the Register Ext PN-Key action (*see screenshot below*). 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.

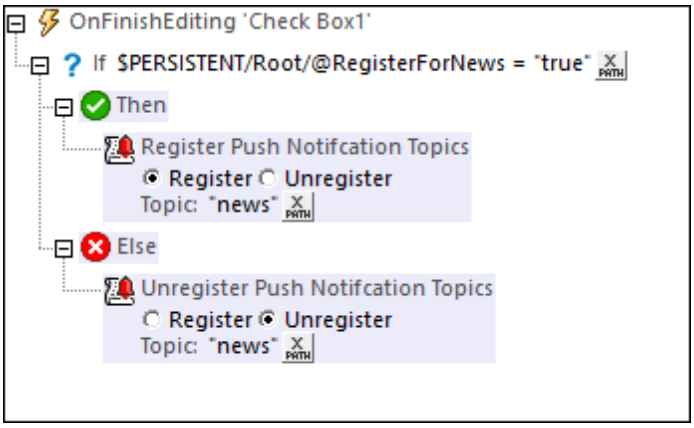


If a PN is sent to a given external PN key, then all devices that are registered with that key will receive the PN.

Note: External PN keys can also be used in AppStore Apps..

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 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.



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 action of the sending solution, iOS PN button sets are defined in the receiving solution via the **Project | iOS Push Notification Button Sets** command.

The receiving solution as an AppStore App

If the receiving solution is being created as an AppStore App, please see Push Notifications in AppStore Apps for additional steps to take.

Push Notifications in AppStore Apps

AppStore Apps are MobileTogether apps that are compiled into different OS-based apps that can be posted to the respective app stores for end users to download. (See the section AppStore Apps 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. 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`** event of the solution has an action defined for it.)

Using PNs in AppStore Apps 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.

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.

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.

---

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.

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 o 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.

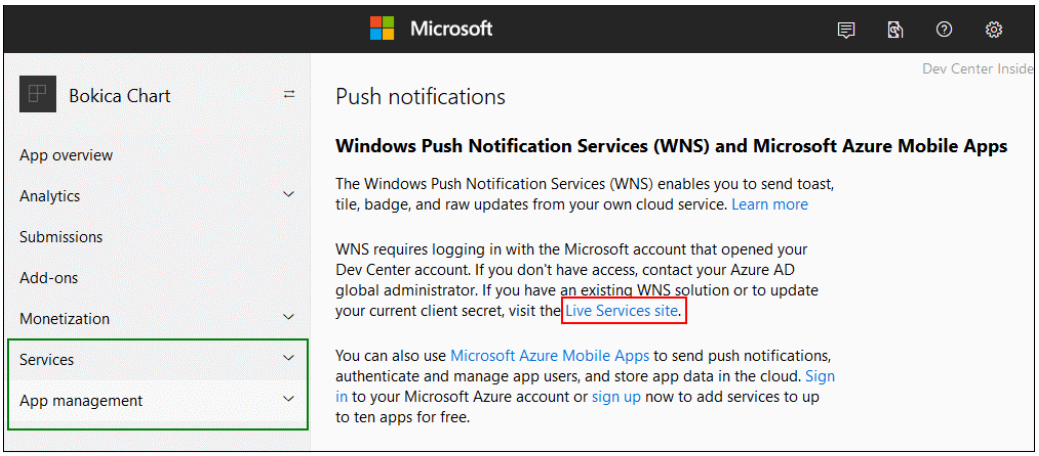
Q: *"Show as notification" does not work on iOS 9.*
A: This is not an error. Support is provided for iOS 10 or later.

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, 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).



4. In the page that appears, click *Live Services site* (*boxed in red in the screenshot below*).
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: (i) Package SID, (ii) Application Secrets. Click **Save** to finish the procedure.
6. You can now start the Generate Program Code Wizard to create the program code.

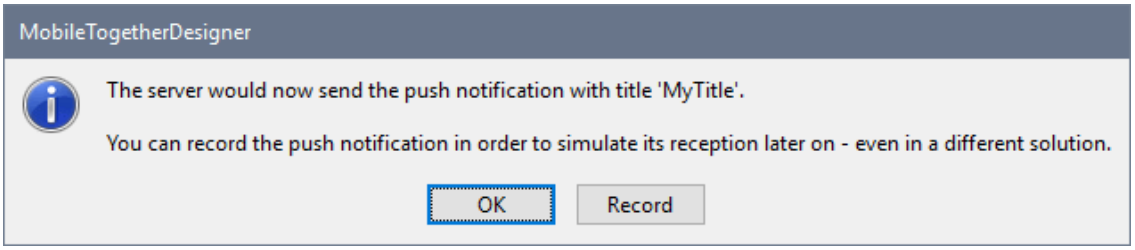
Simulating Push Notifications

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.



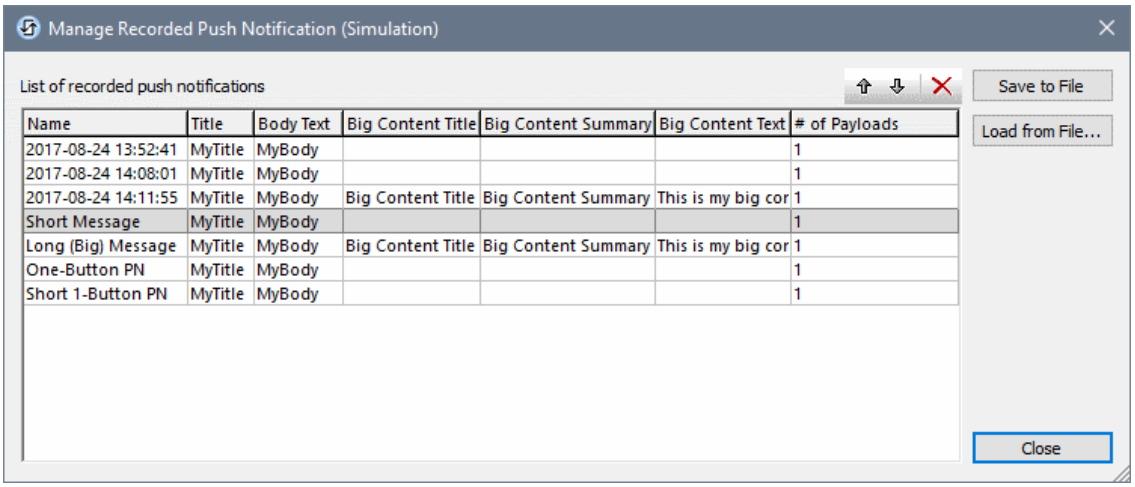
Click **Record** to record the PN data to memory. You can record multiple PNs to memory in this way. On closing the simulation, the recorded PNs are in memory—but not yet saved to file.

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.

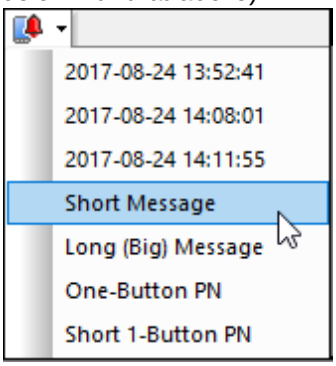
Loading recorded PN Simulation Data into the simulator

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.



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.

11.6 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 explains how to select data for the different axes
- Chart Settings and Appearance describes how to define the properties of charts

Creating and Configuring Charts

This section:

- Creating a chart
- The context node
- The Chart Configuration dialog
- Editing chart settings and data selection

Creating a chart

To insert a chart in the design, drag the Chart control from the Controls Pane to the location in the design where you wish to insert the chart.

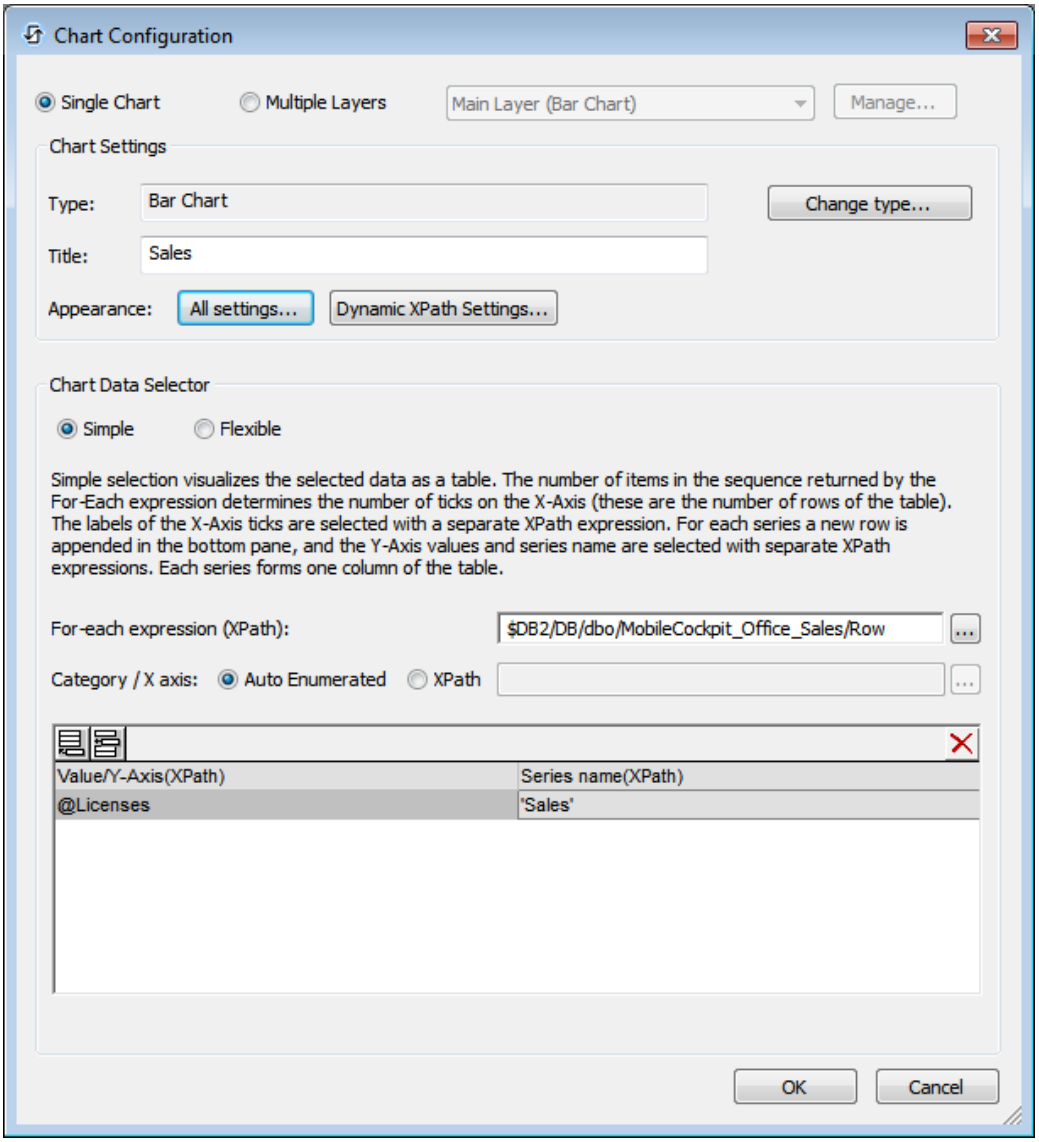
The context node

Drag an XML node from the Page Sources Pane 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, and (ii) defining the chart's properties. 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



The Chart Configuration dialog has three parts:

- *Single or Multiple layers:* Multiple layers can be selected to create overlay charts; charts are overlaid one upon the other to produce a composite
- *Chart Settings:* To select the chart type and define the appearance of the chart
- *Chart Data Selection:* To select the data for the various axes of the chart using either the simple option or flexible option

---

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 **Edit Chart Settings**. This pops up the Chart Configuration dialog (*screenshot above*) of that chart. You can edit the chart's settings or data

selection in the dialog, then click **OK** to finish.

Chart Data Selection

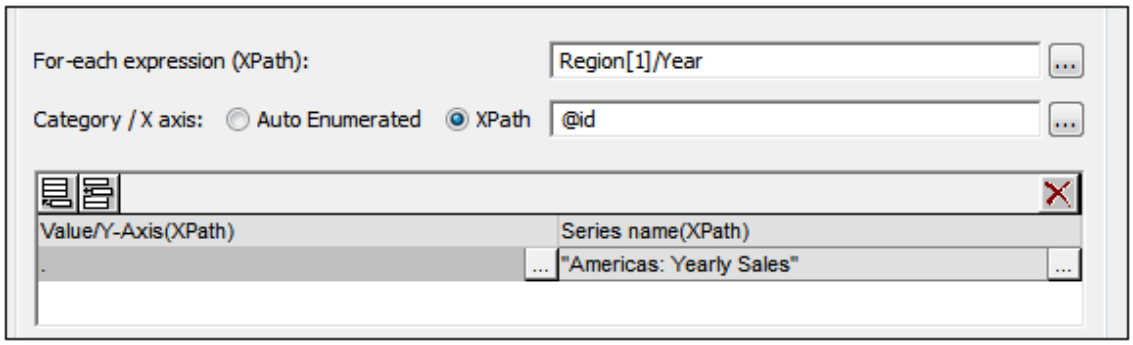
This topic contains simple examples to illustrate how chart data selection works.

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">
       <ChartType>Pie Chart 2D</ChartType>
       <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

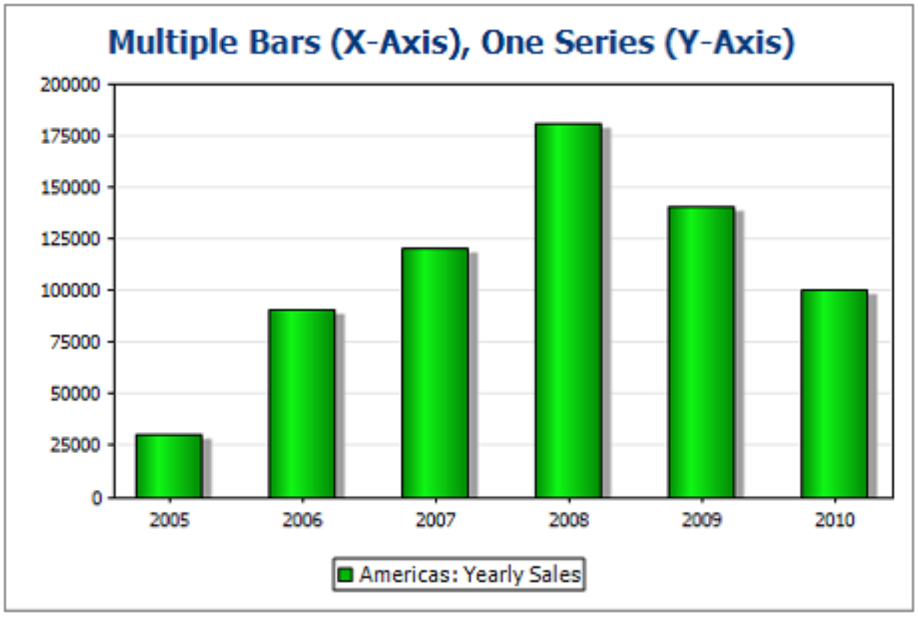
The screenshot below shows the Chart Configuration dialog, at the bottom of which is the Chart Data Selector pane with fields for entering the four XPath expressions for data selection.



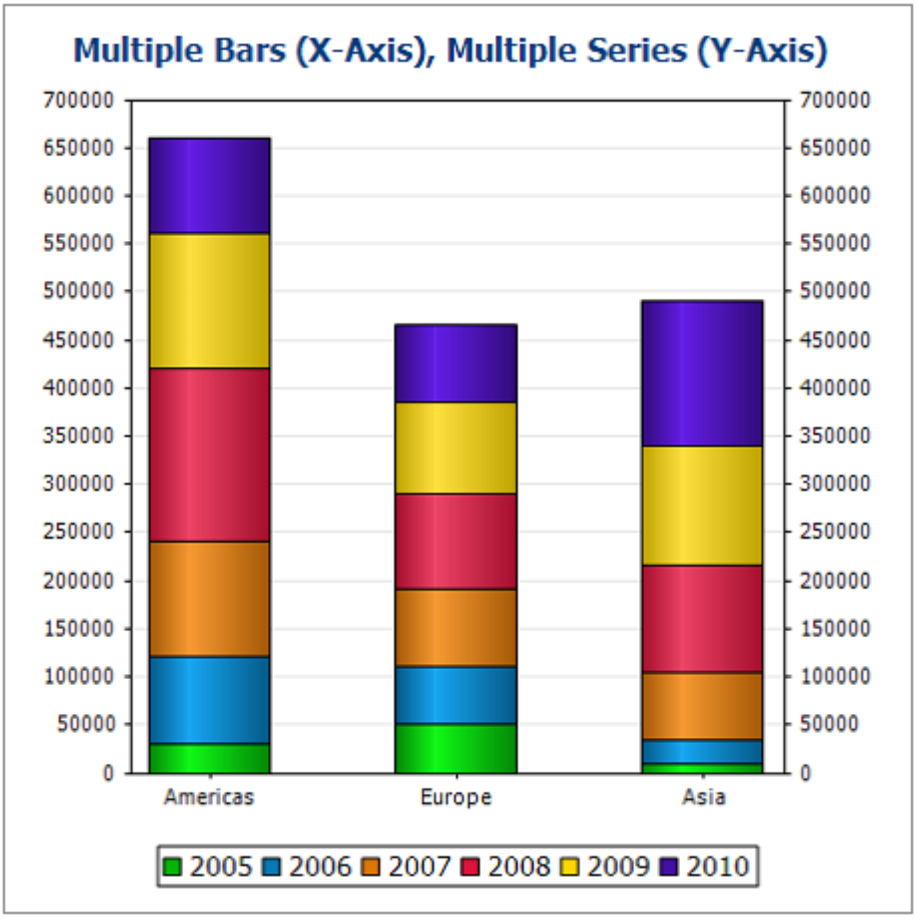
The four XPath expressions in the Chart Data Selector pane work together and do the following:

XPath	Description
For-Each	• Sets the context for the other three XPath expressions • Sets the number of items in the returned sequence as the number of ticks on the X-Axis. • 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. • 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*). • 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. • Each series is defined by two XPath expressions: one for the series value, the other for the series name. • 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*).

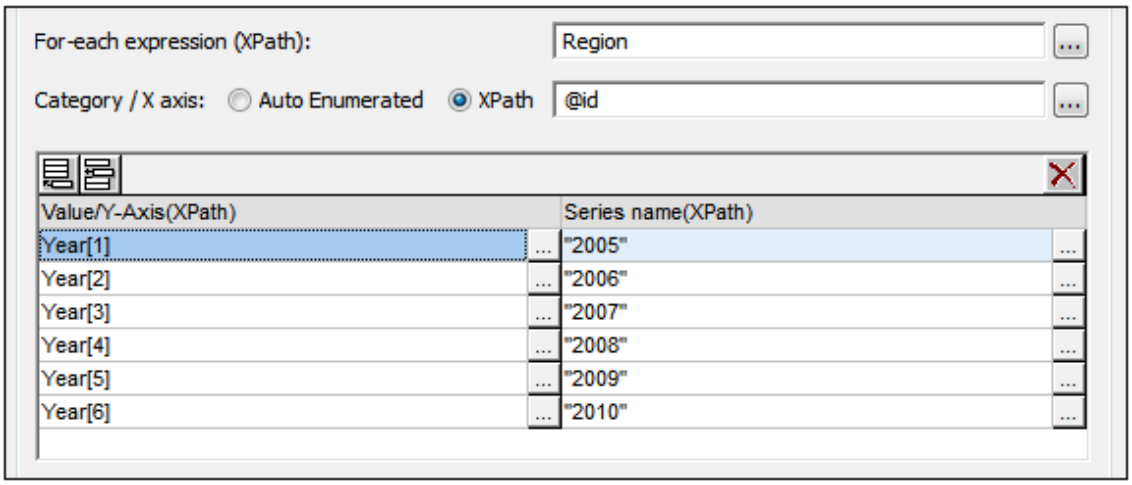
A bar chart that is generated for the data selection shown in the screenshot above and the XML data in `YearlySales.xml` looks like the chart in the screenshot below.



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.



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.

Chart Data Selection: Simple

This section:

Introduction

In the Chart Data Selector pane of the Chart Configuration dialog, 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">
       <ChartType>Pie Chart 2D</ChartType>
       <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>
```

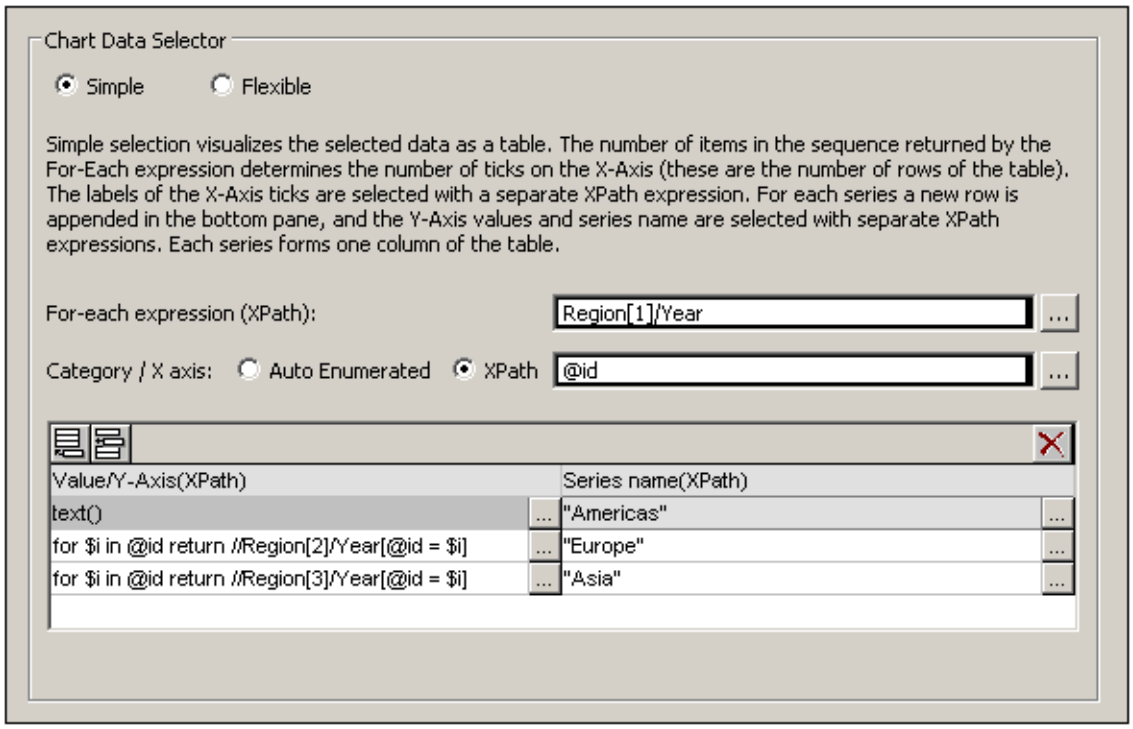
The context node

In the design, drag an XML node from the Page Sources Pane 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.



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.

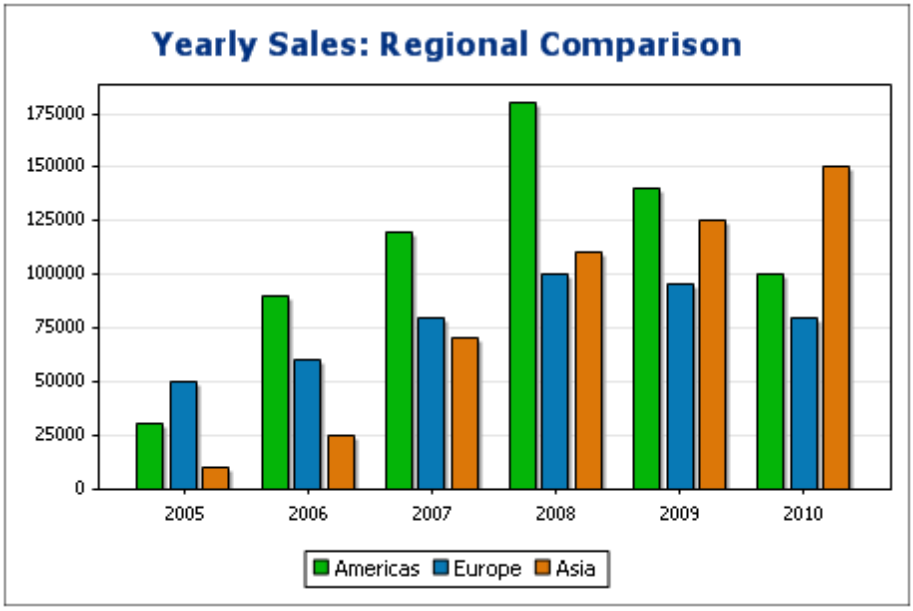
For-Each XPath	X-Axis	Y-Axis for Series		
		Americas	Europe	Asia
`Region[1]/Year[1]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[2]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[3]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[4]`	`@id`	`text()`	`XPath-1`	`XPath-2`
`Region[1]/Year[5]`	`@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

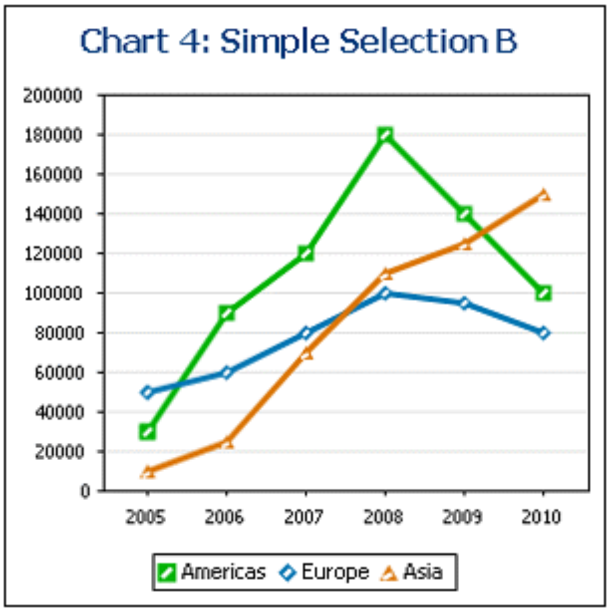
the table). The number of items in the sequence returned by the For-Each expression determines the **number of ticks** on the X-Axis.

- 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:

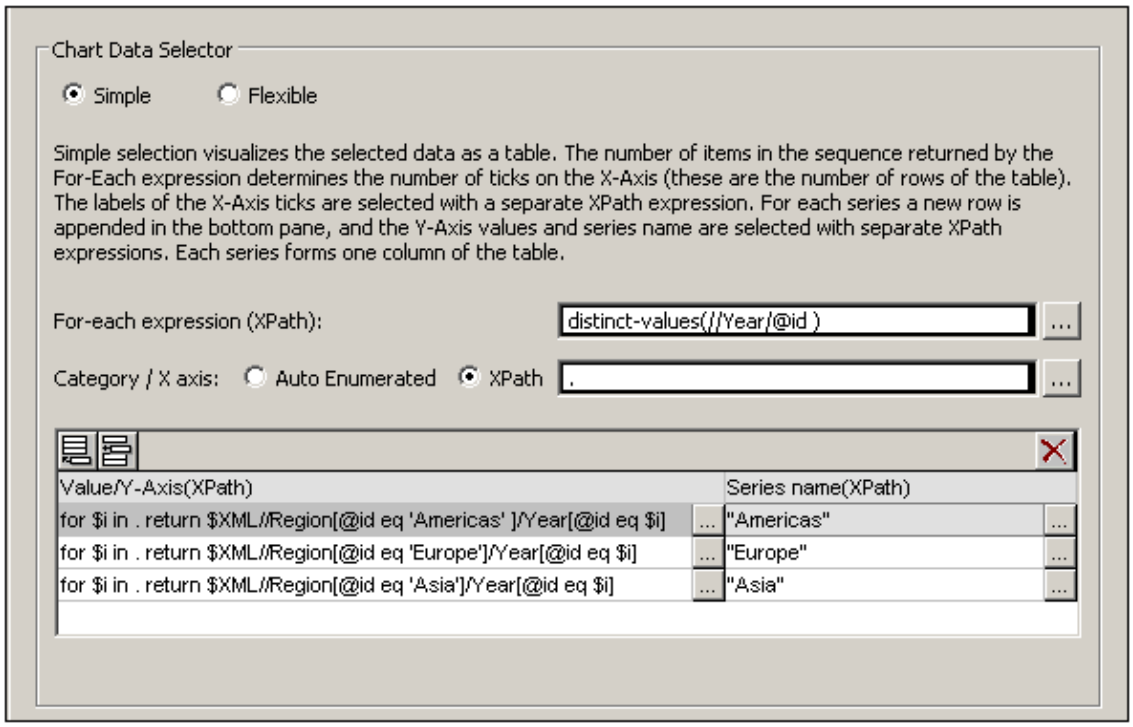


The line graph chart for this data selection would look like this:



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.



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:

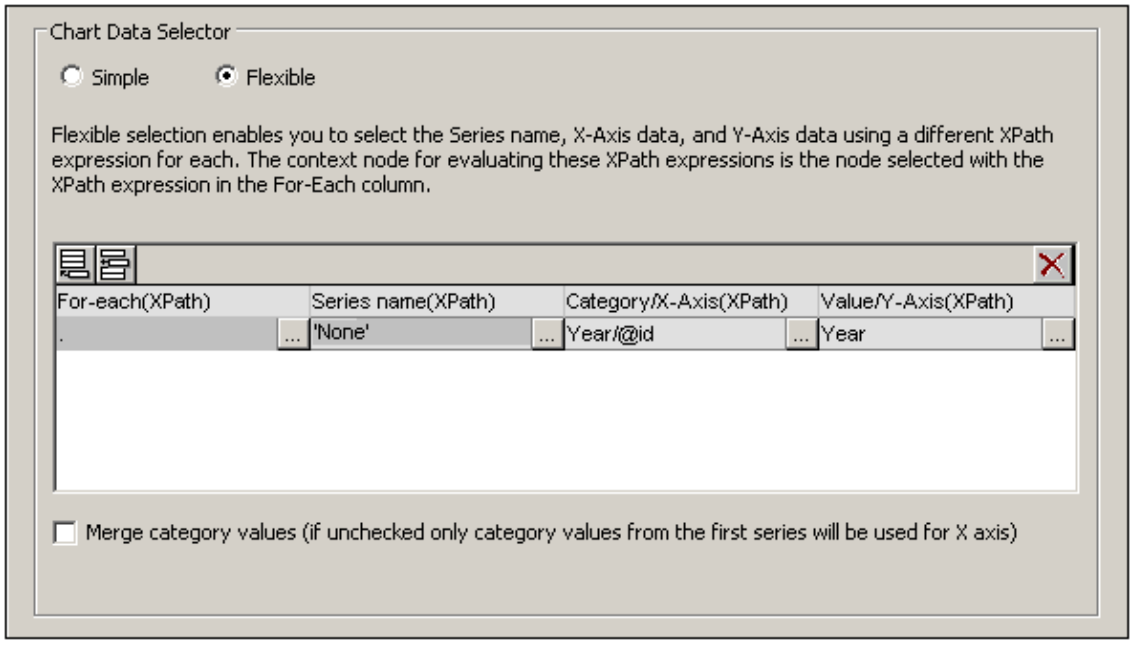
For-Each XPath	X-Axis	Y-Axis for Series		
		Americas	Europe	Asia
2005	2005	XPath-1	XPath-2	XPath-3
2006	2006	XPath-1	XPath-2	XPath-3
2007	2007	XPath-1	XPath-2	XPath-3
2008	2008	XPath-1	XPath-2	XPath-3
2009	2009	XPath-1	XPath-2	XPath-3
2010	2010	XPath-1	XPath-2	XPath-3

Chart Data Selection: Flexible

This section:

About flexible chart data selection

In the Chart Data Selector pane (*screenshot below*) of the Chart Configuration dialog, 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.



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 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.

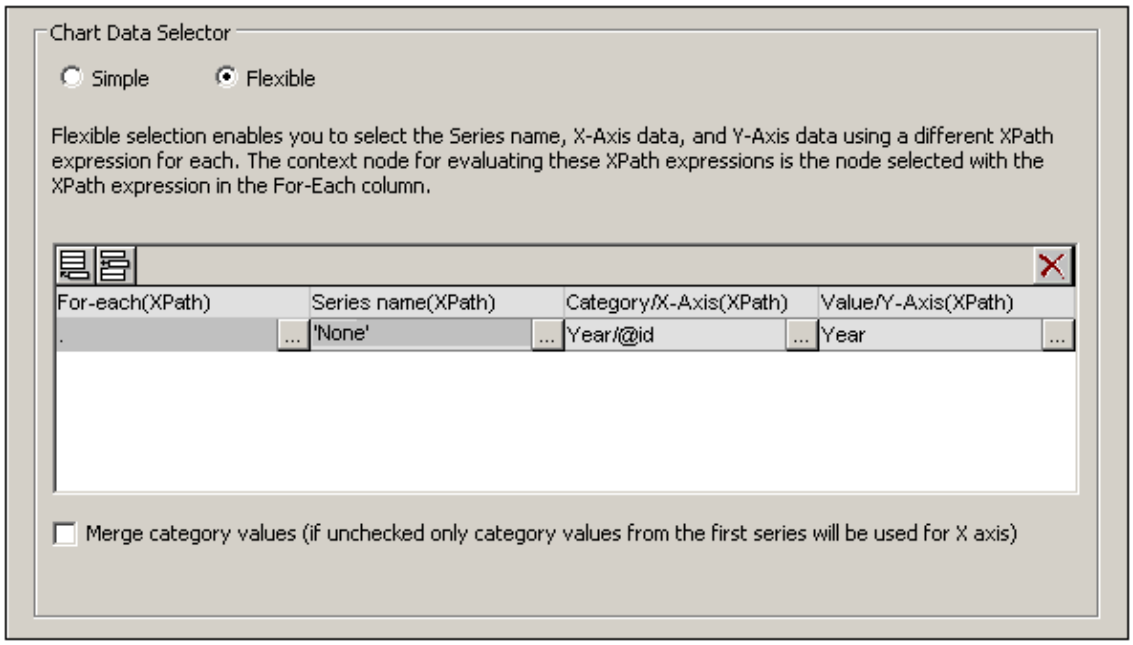
The following examples illustrate important points to consider when selecting data for the axes. They reference the XML document listed below.

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">
       <ChartType>Pie Chart 2D</ChartType>
       <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>
```

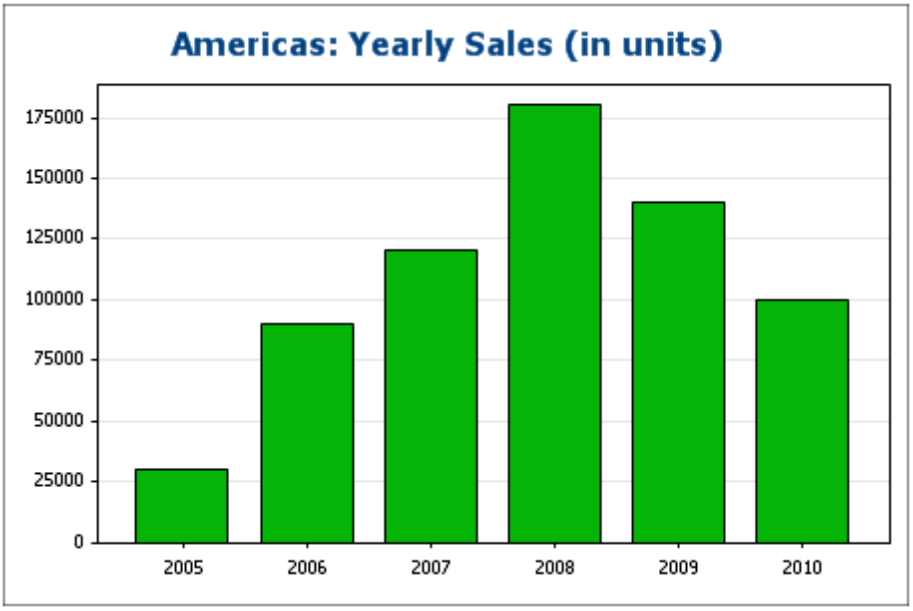
One row, one series

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.



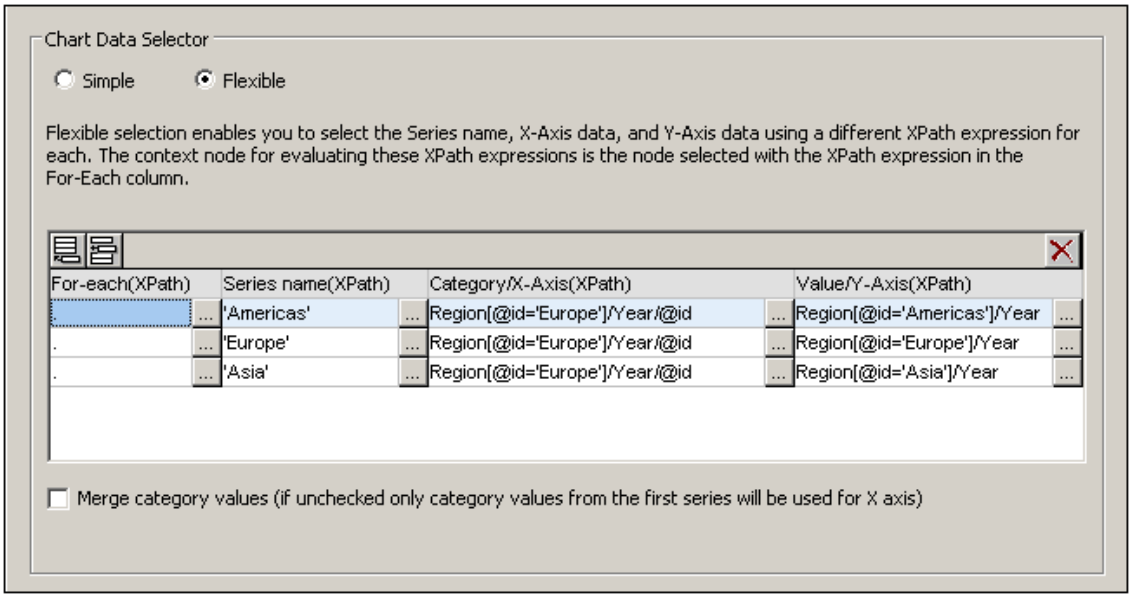
In the chart data selection shown in the screenshot above, the For-Each expression returns the current node (which is the `Region` element), so the `Region` element will be the context node for all the other three XPath expressions (series, X-Axis, and Y-Axis). Since there is only one series in this 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.



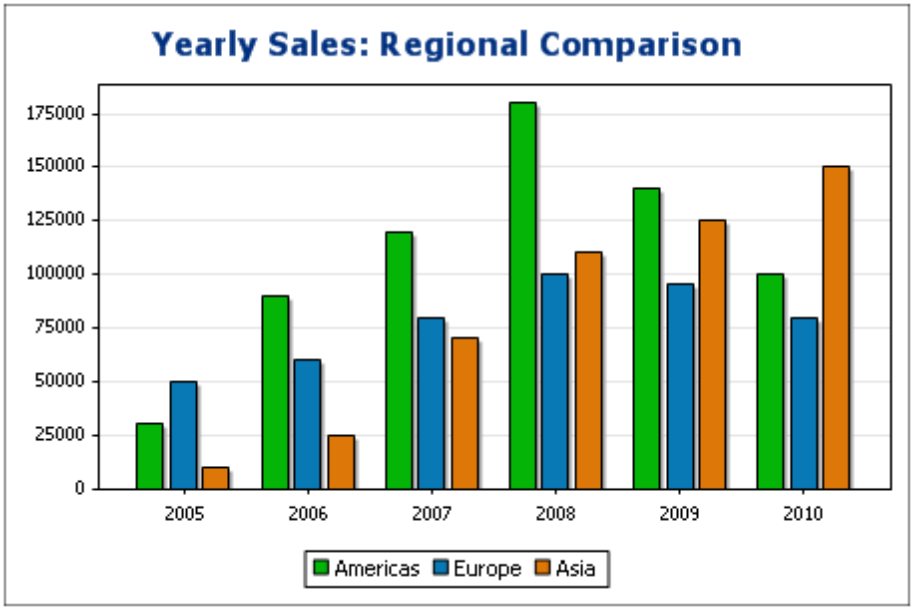
The important points to note about the data selection above are:

- 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).

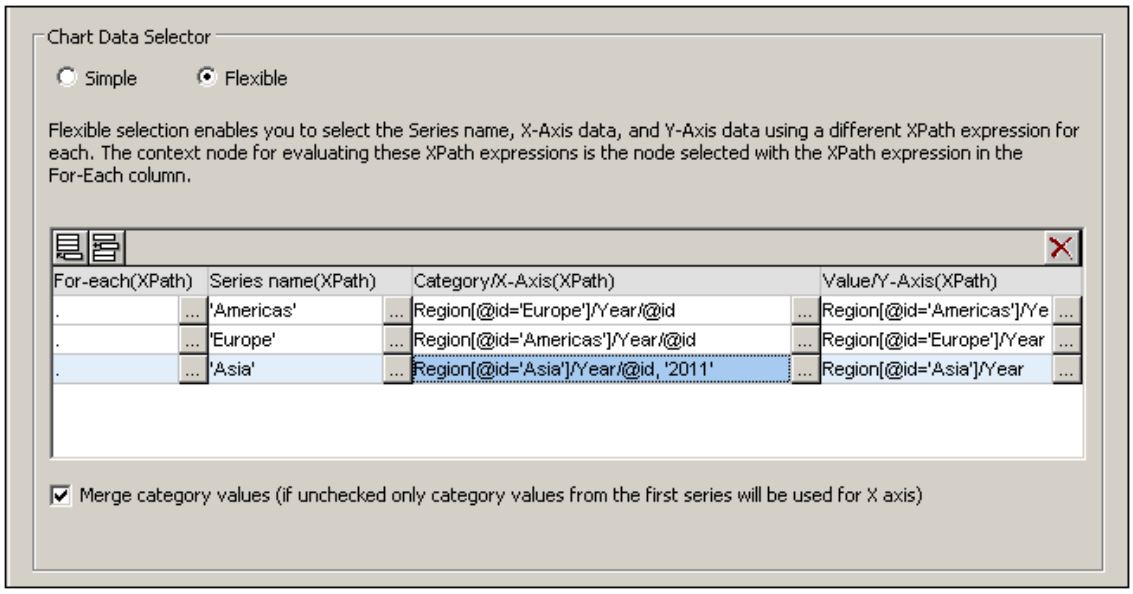
- The first row is set to define the Americas series and is given a string expression as its series name. The X-Axis values are selected using the `Year/@id` values of the Europe region (it doesn't matter which region is selected since all have the same `Year/@id` values). The Y-Axis values of the first (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:

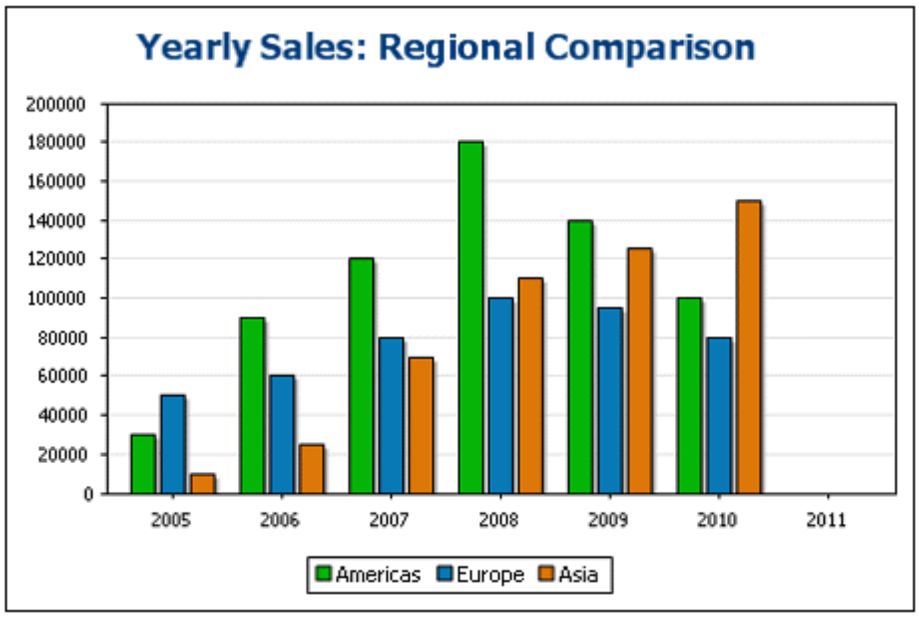


Three rows, three series, category values merged

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 to `20000`.

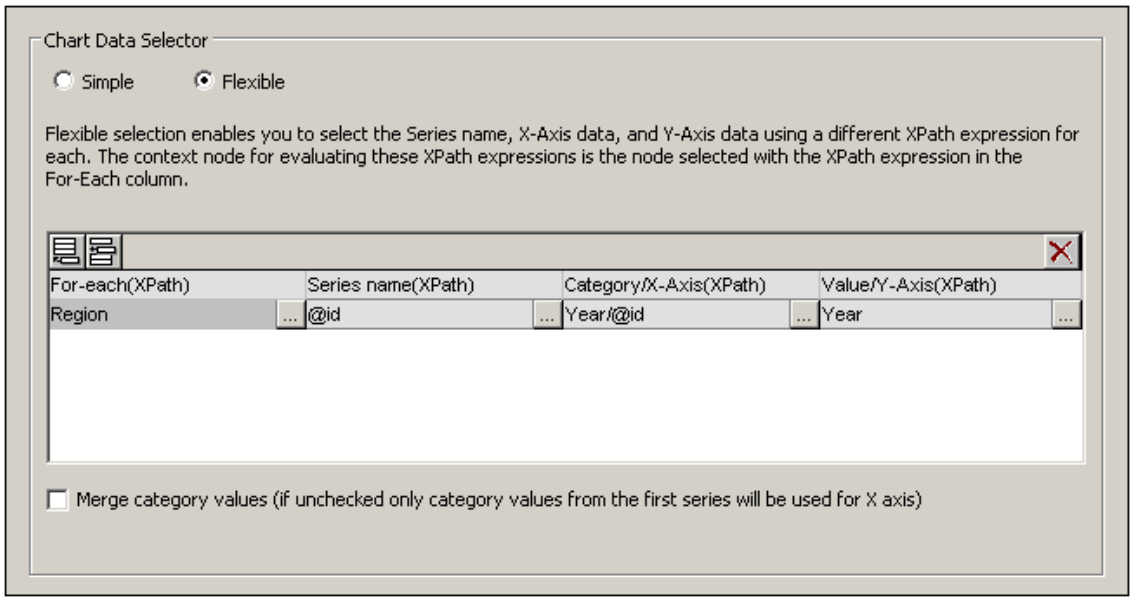


The effect of this change is to add one new item (`2011`) to the X-Axis result sequence. The chart would look something like this:

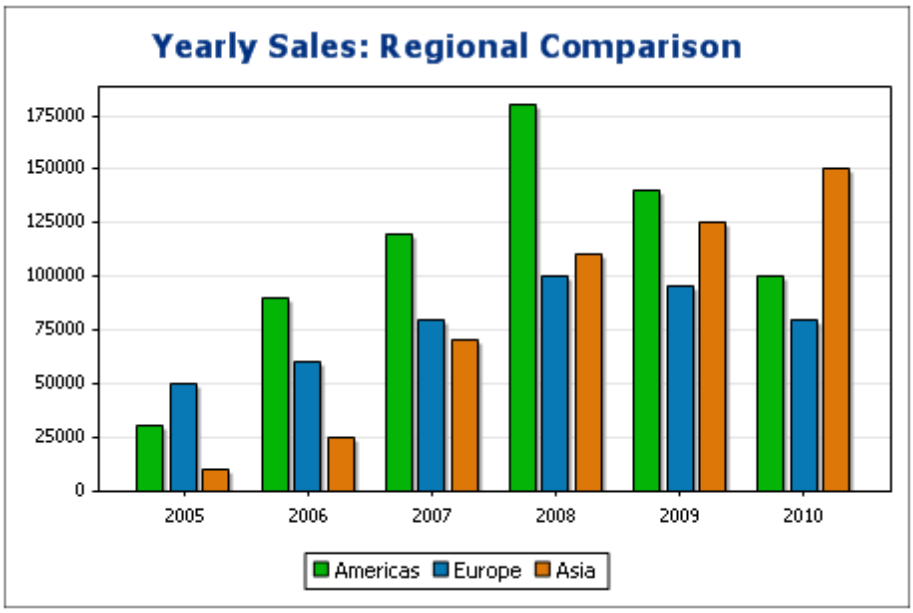


One row, three series

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.



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

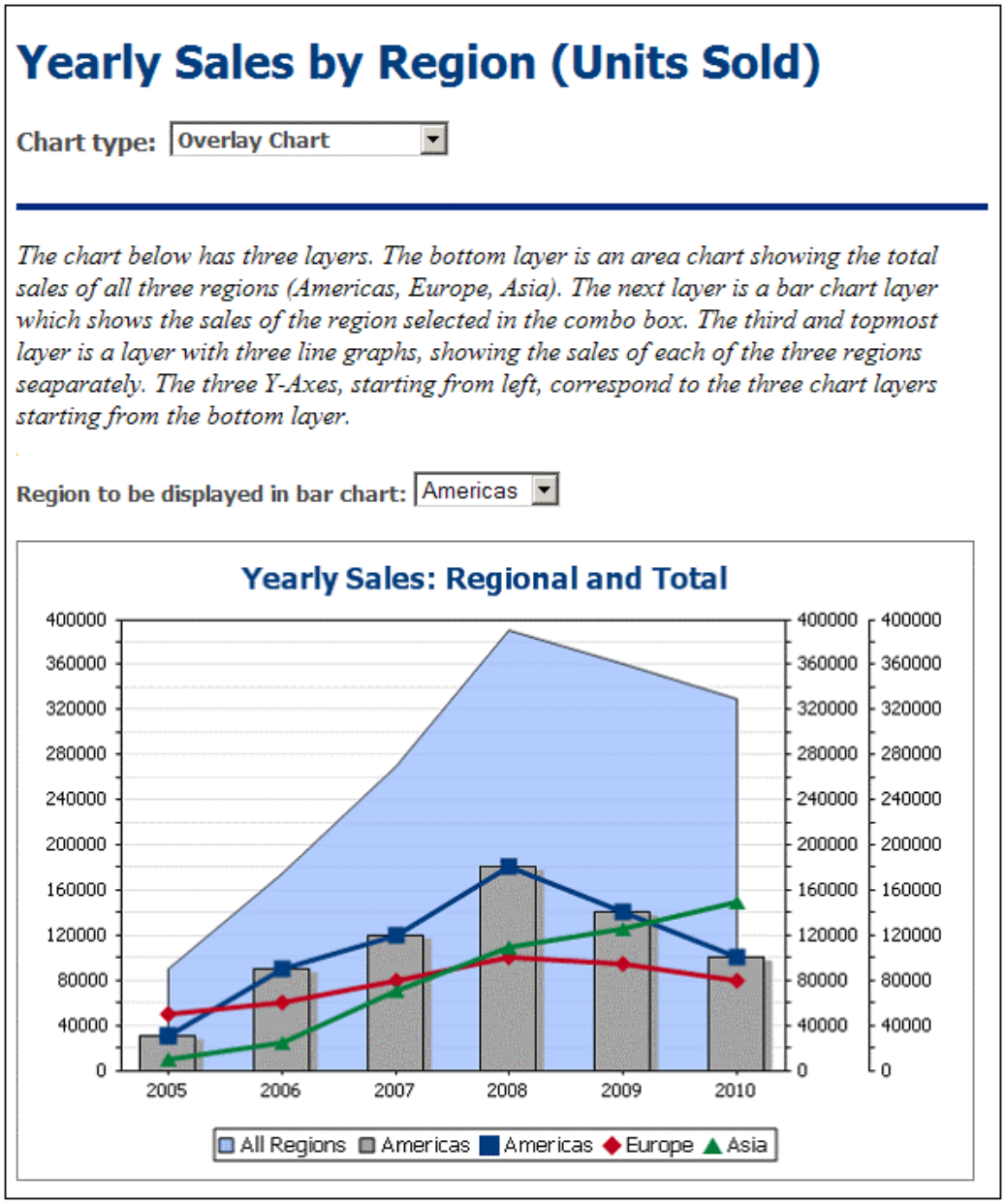
The following points should be noted when using the Chart Data Selector to select data for the various chart axes:

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.

Overlay Charts

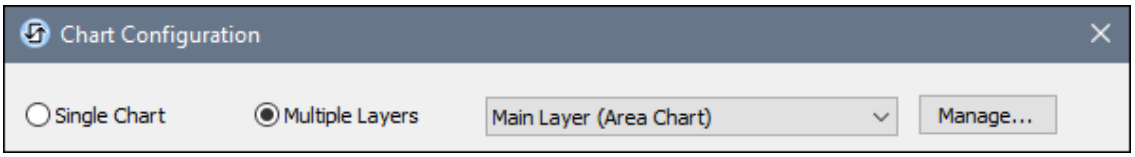
Overlay charts (*screenshot below*) enable you to combine multiple charts of different types. The overlay chart in the screenshot below is explained in the text of the screenshot. The following types of chart can be combined:

- 2D bar charts and stacked bar charts
- Line charts
- Area charts and stacked area charts
- Candlestick charts



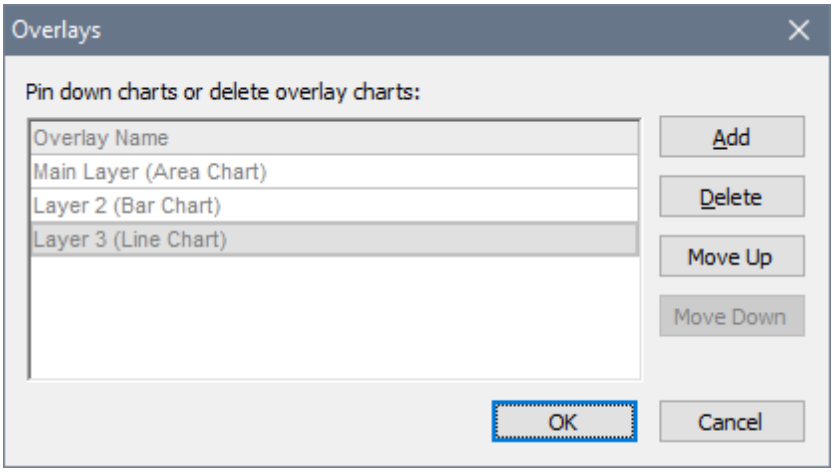
Creating the overlay chart

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 and Chart Data Selection: Flexible 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

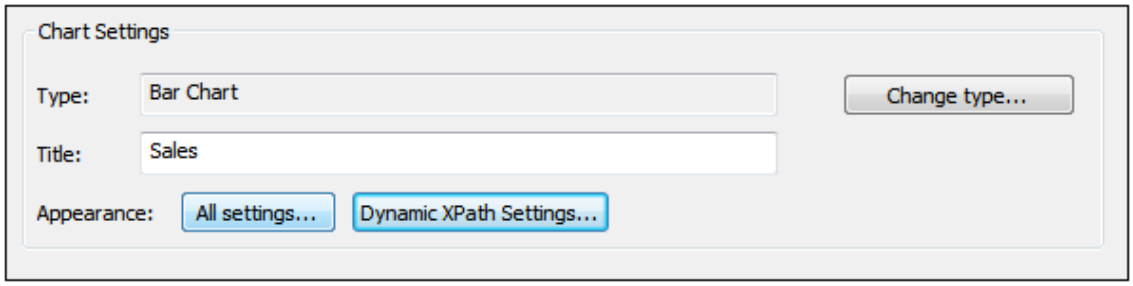
The following chart settings are relevant:

- 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.

Chart Settings and Appearance

Chart settings are organized as follows:

- Basic Chart Settings, 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,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. To access this dialog, click **All Settings** in the Chart Configuration dialog (*see screenshot below*).
- Dynamic XPath Settings, which can be accessed by clicking **Dynamic XPath Settings** in the Chart Settings pane (*see screenshot below*).



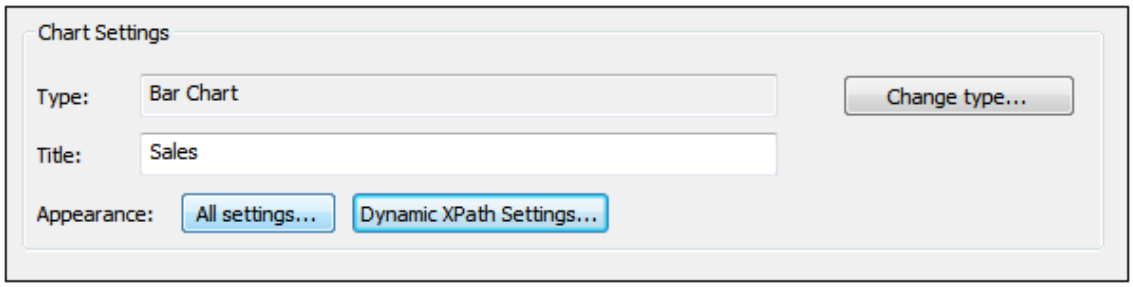
Basic Chart Settings

This section:

- Setting the chart type
- List of chart types
- Other basic settings

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 of the Chart Configuration dialog.



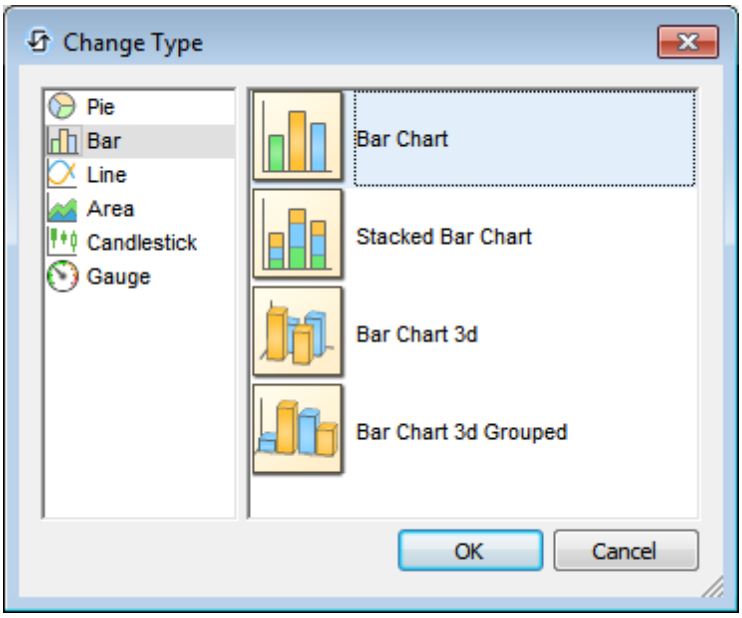
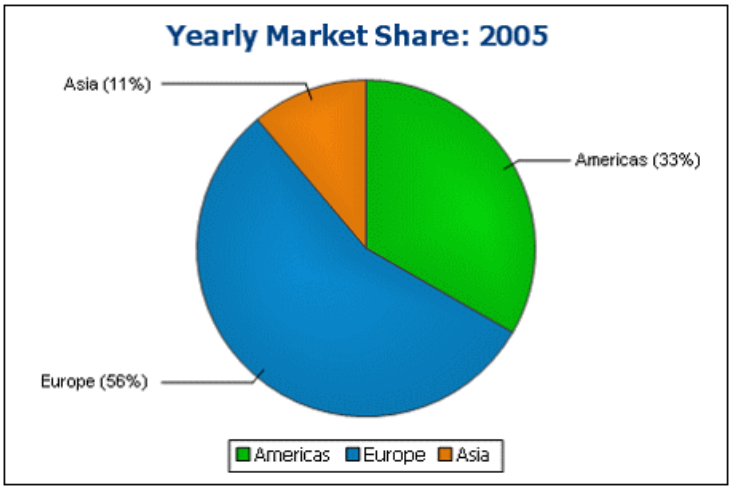


Chart types

The various types of charts that are available are listed below. In the Change Type dialog (*screenshot above*), select the chart type you want and click **OK**.

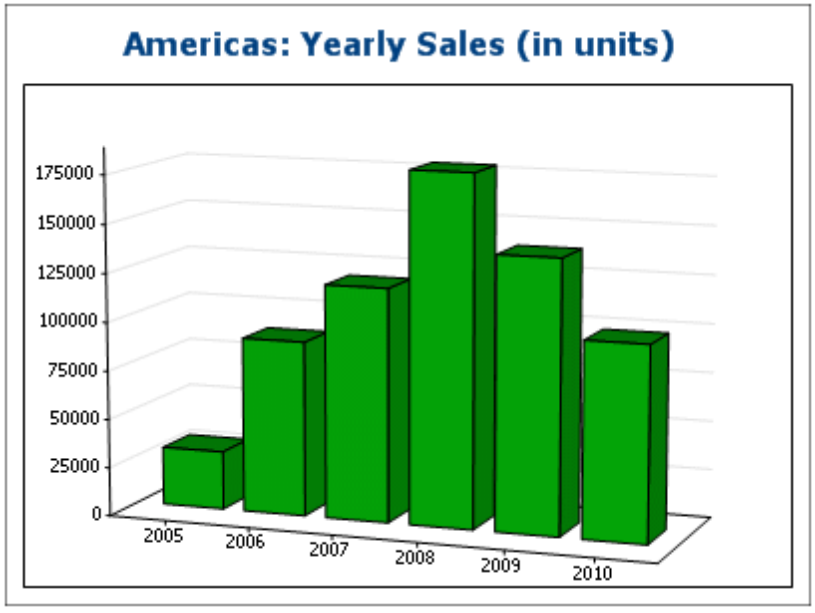
- *Pie charts*

  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.

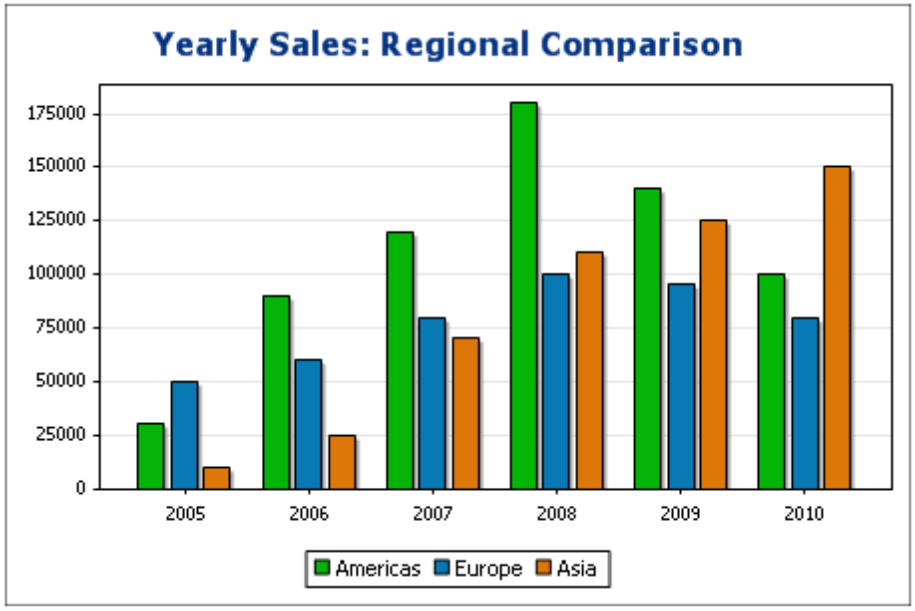


- *Bar charts*

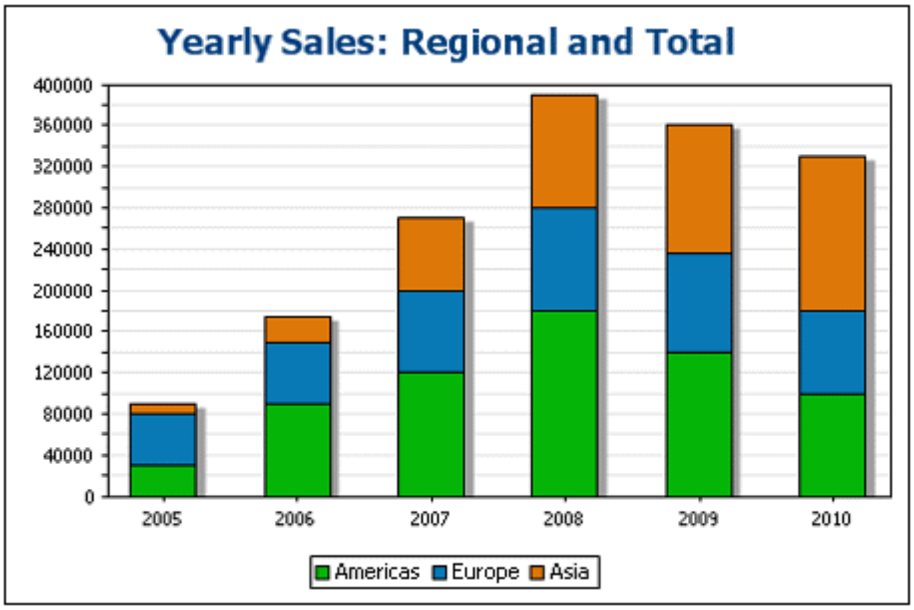
  Bar charts can have two sets of values used along two axes (*below*).



  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*).

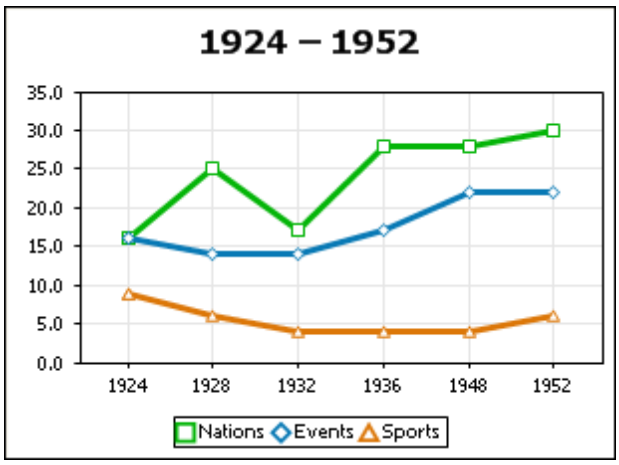


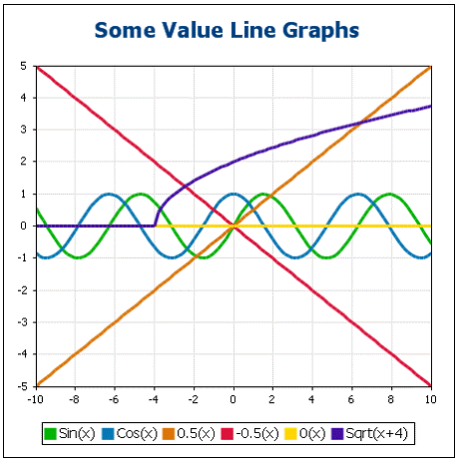
A three-axis bar chart can also be stacked if you need to show totals. Compare the stacked chart below with the chart above. The stacked chart shows the total of sales on all continents.



- *Line charts*

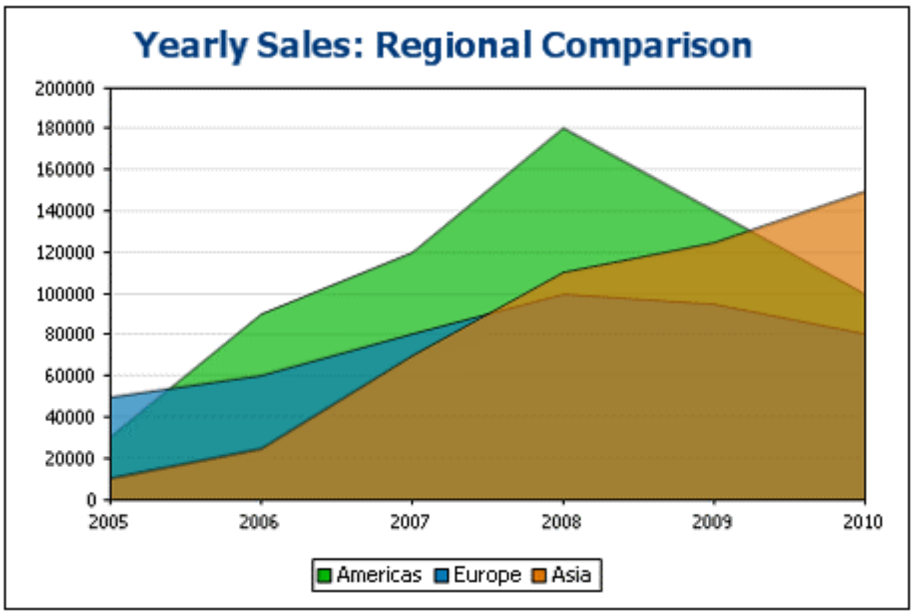
  The difference between a line chart (*below left*) and a value line chart (*below right*) is that value line charts only take numerical values for the X-axis. If you need to display line charts with text values on the X-axis, use line charts.





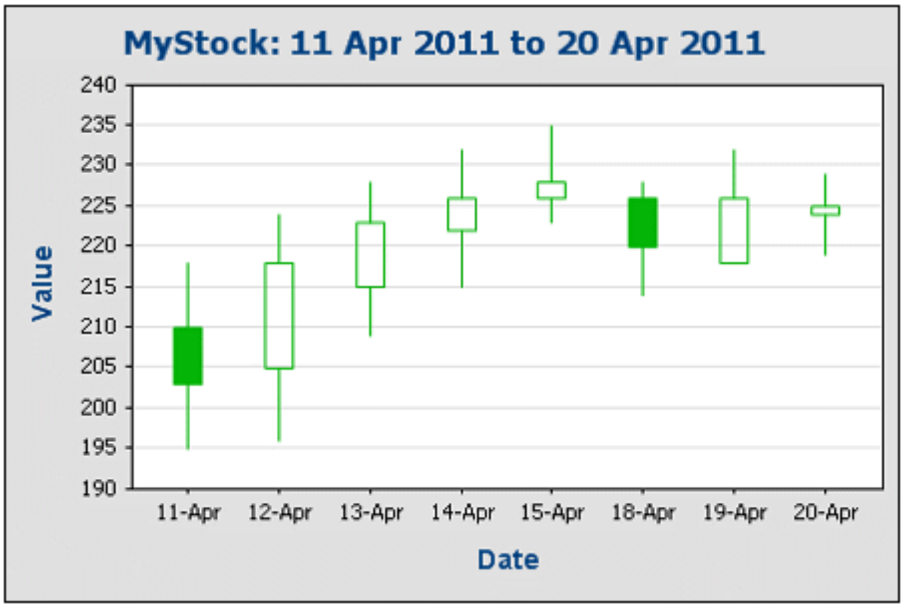
- *Area charts*

  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*).



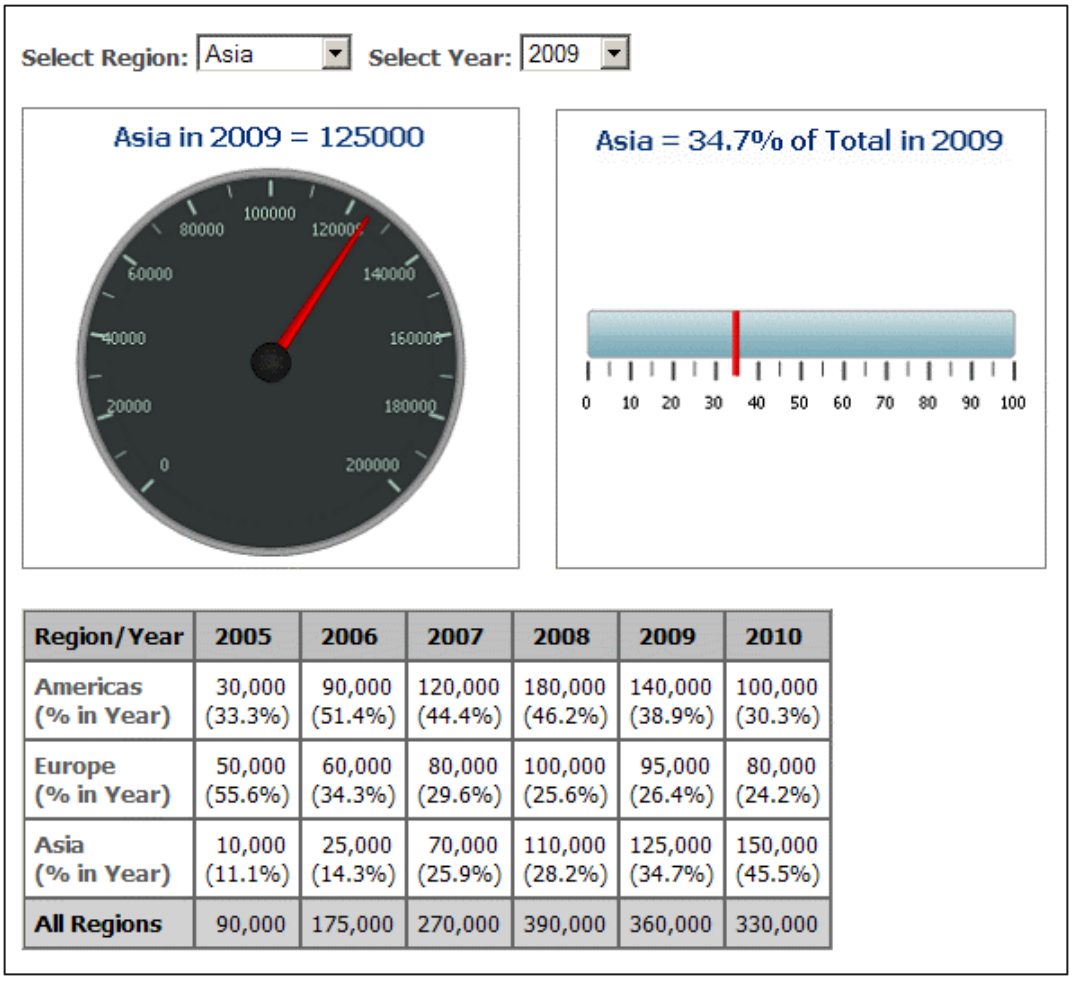
- *Candlestick charts*

  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.



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*).

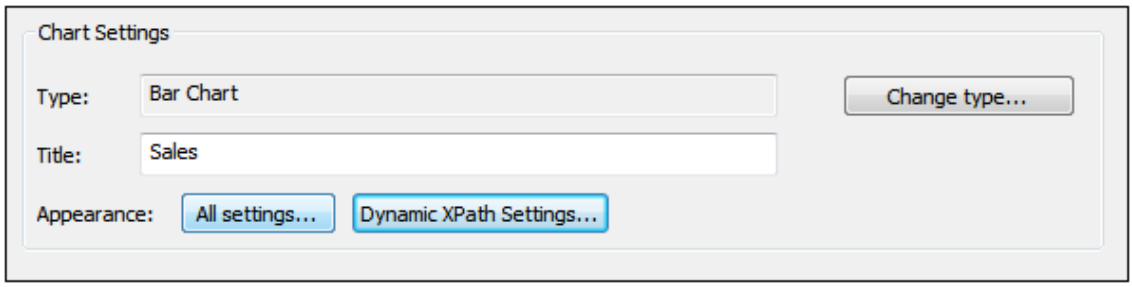


Chart Settings
Type:
Bar Chart
Change type...
Title:
Sales
Appearance:
All settings...
Dynamic XPath Settings...

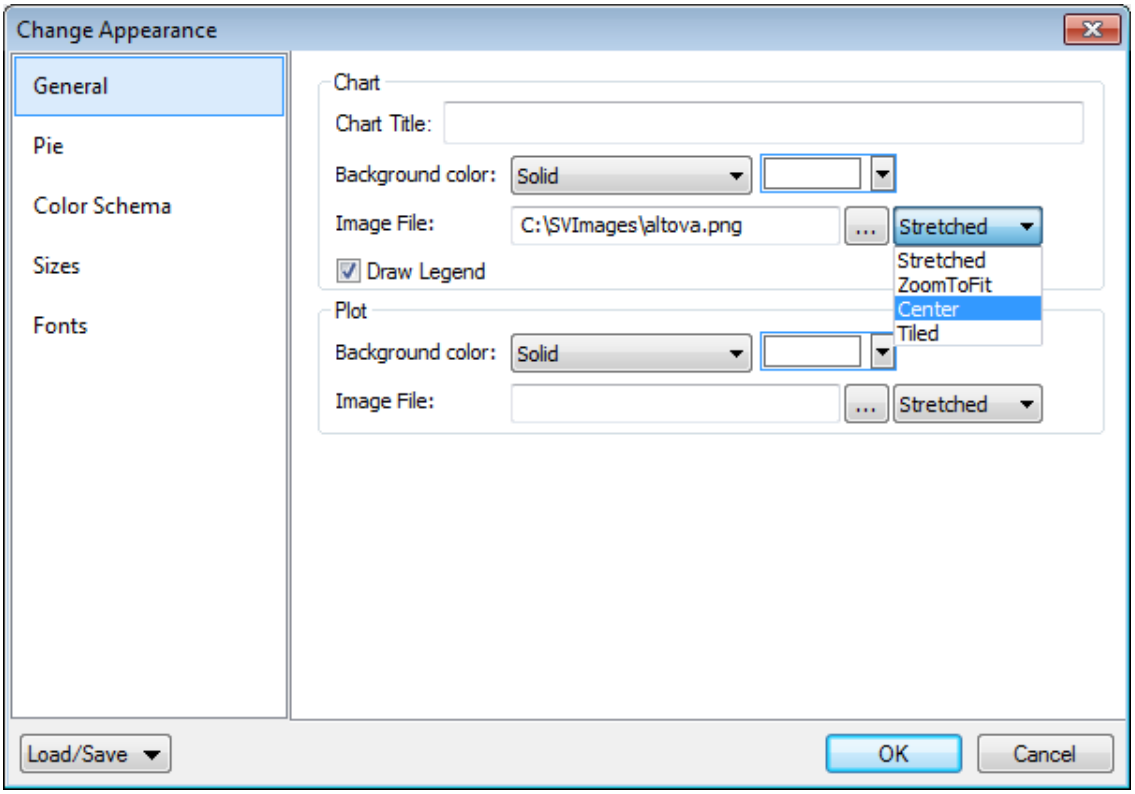
Advanced Chart Settings

This section:

- Accessing the advanced settings
- Overview of advanced settings
- Loading, saving, resetting chart settings

Accessing the advanced settings

To access a chart's advanced settings do the following: Click **All Settings** 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*).



Overview of advanced settings

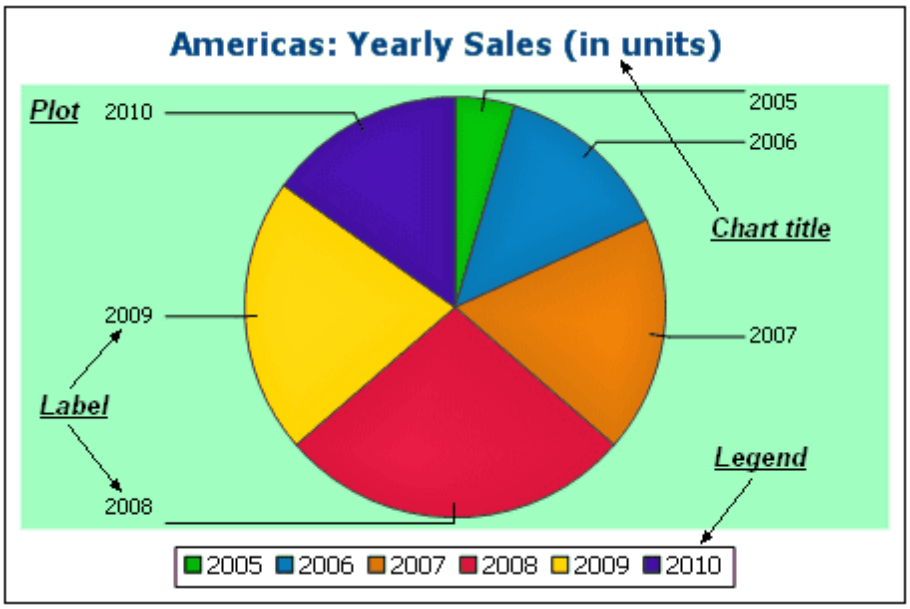
The advanced settings are organized into tabs that are common to all chart types and those that are specific to a single chart type.

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.

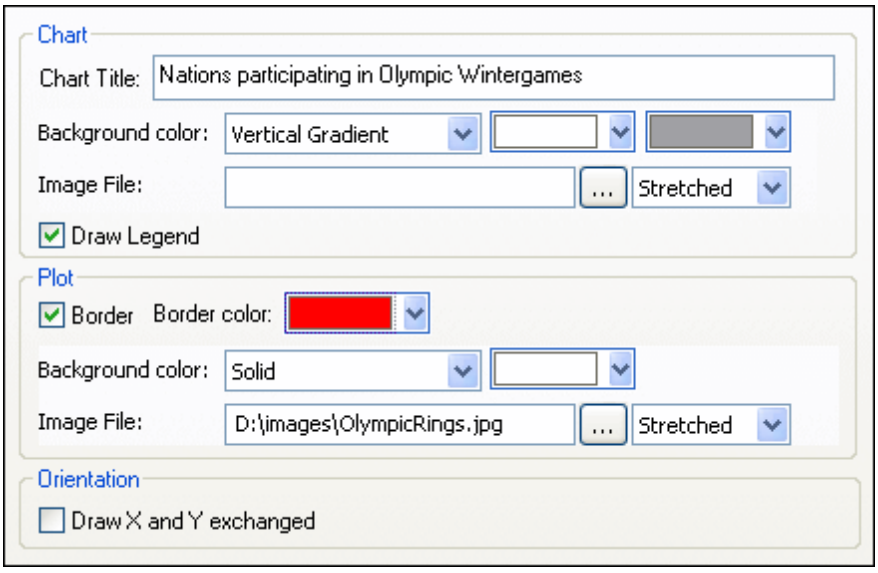
Loading, saving, resetting chart settings

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) 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.

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

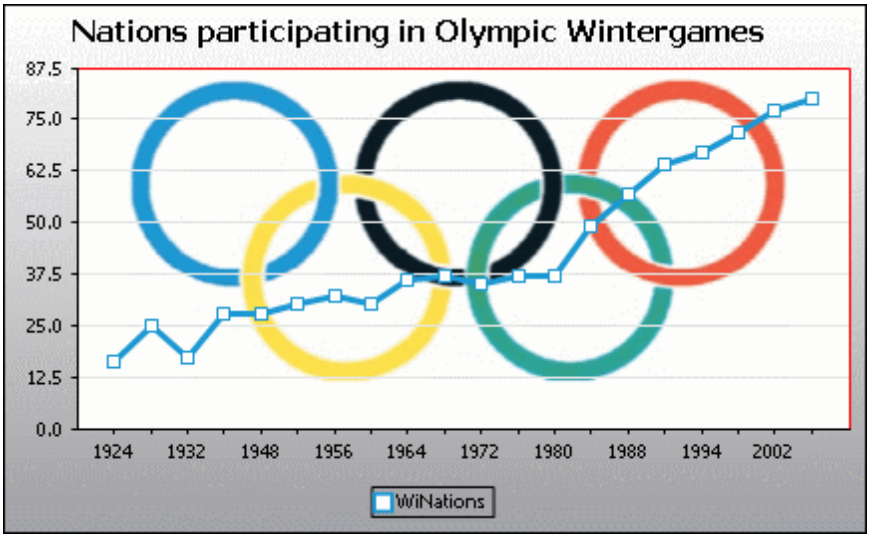
Enter a descriptive title for your chart into the `Chart Title` field and select a background color for the entire chart from the drop-down list. You can choose a solid background, vertical gradient, or horizontal gradient and define start and end colors for the gradient, if applicable. In addition, 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.

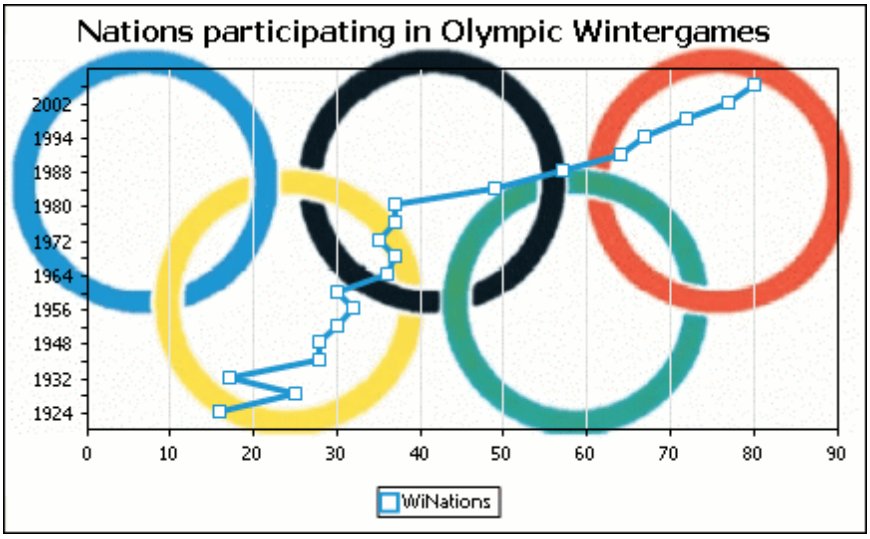
Plot

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.



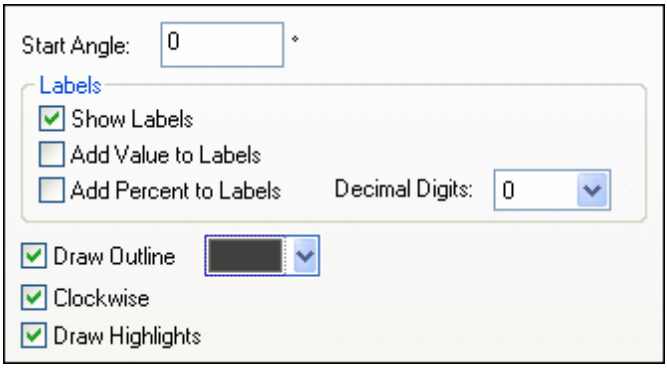
Note that this option is not available for pie and gauge charts.

Type-Related Features

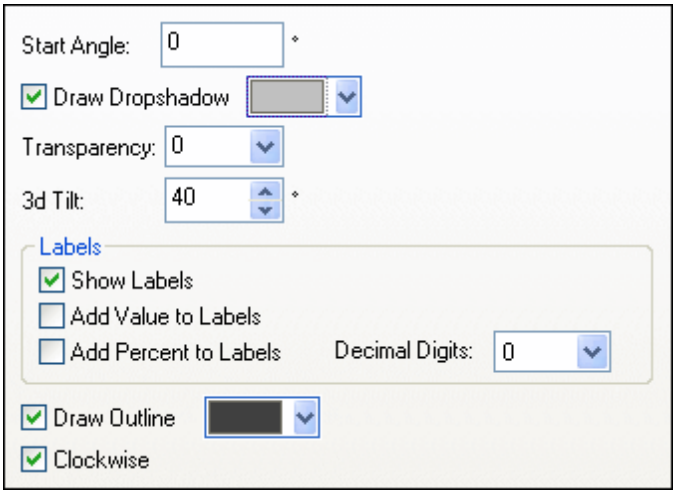
For each of the chart types, and even for the various sub-types, the **Change Appearance** dialog box provides a section where you can define the type-related features of the chart.

Pie chart

Most settings are the same for the 2d and 3d versions. In 2d pie charts, you can additionally draw highlights.

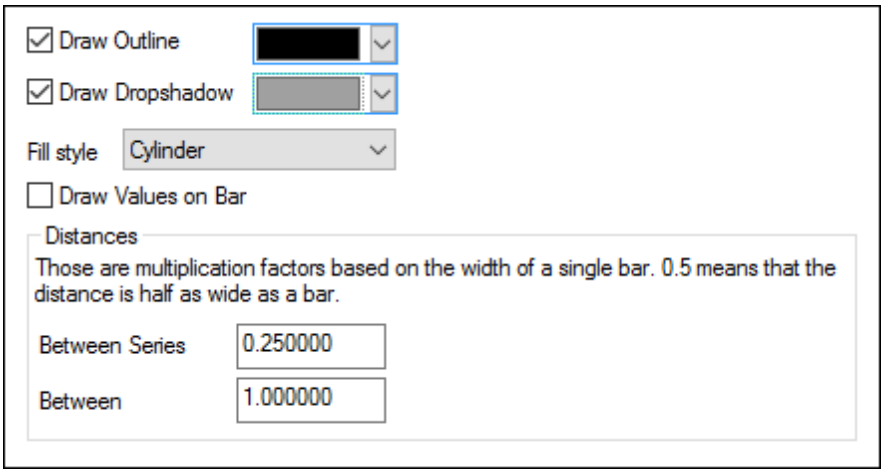


In 3d pie charts, you can display drop shadows, add transparency and define the 3d tilt.



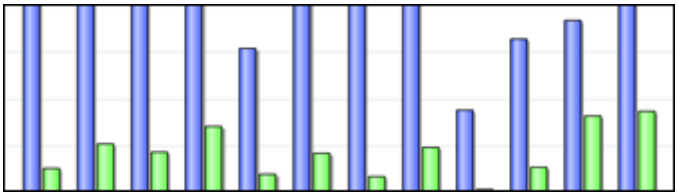
The `Start Angle` value defines where the first row of the selected column will be displayed in the chart. An angle of 0 degrees corresponds to 12 o'clock on a watch.
You can show labels in addition to, or instead of, the legend, add values and/or percentage to the labels, and define for the percentage 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

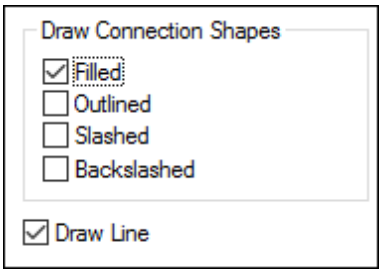


For bar charts, you can make the following setting:

- 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.
- 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.
- 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.



Line chart

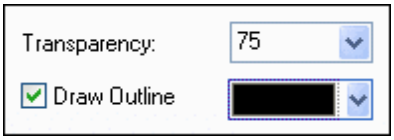


To draw connection shapes that mark the values in line charts, you need to activate at least one check box in the Draw Connection Shapes group box. You can use five different shapes to mark a series: square, rhomb, triangle, inverted triangle, and circle. If there are more than five series in your chart 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.

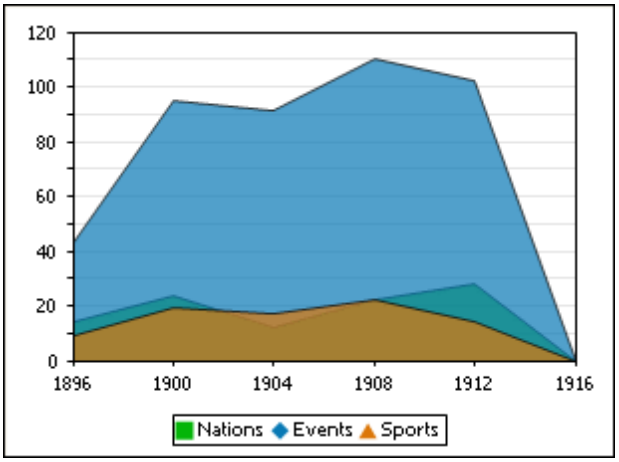
The *Draw Line* option enables the graph to be drawn with (i) only connection shapes, or (ii) with connection shapes joined by a line.

Connection shapes are available for both line charts and value line charts.

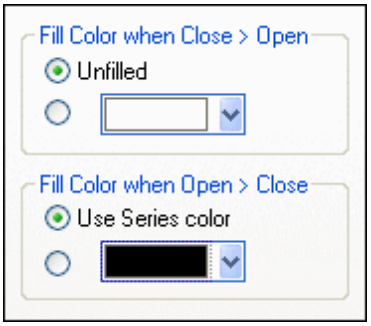
Area chart



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 that the opening value.

Gauge chart

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.

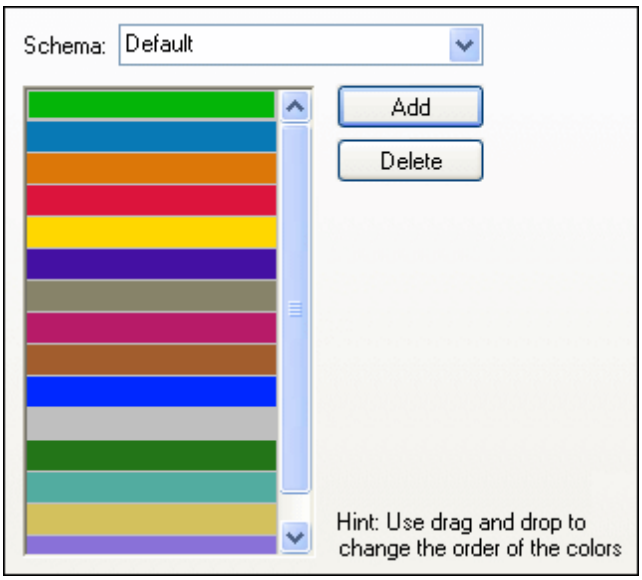
Colors

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

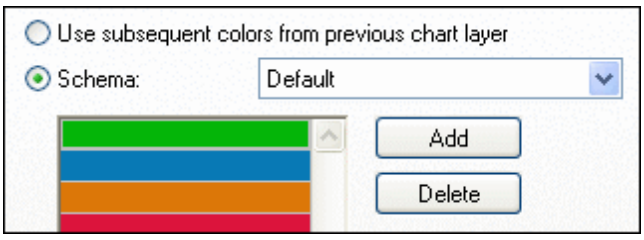
Color Schema

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.



The top color will be used for the first series, then the second color and so on. You can change the order or the colors by selecting a color and dragging it to its new positions 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.



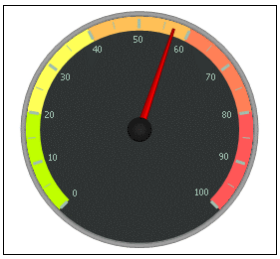
When the radio button is activated, the color schema from the previous layer will be used and you cannot choose a separate color schema for the overlay. The series of the active layer will be displayed using subsequent colors from the color schema of the previous layer. This way, all series of the Charts window have different colors and can therefore be distinguished more easily.

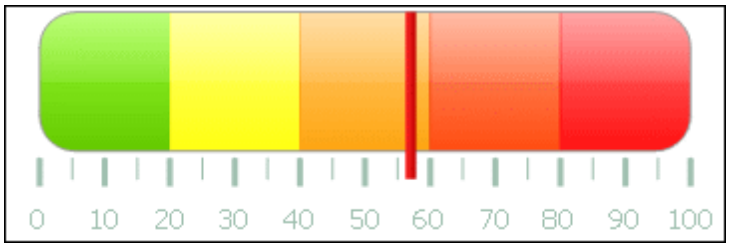
You can break this link on any additional layer that you add and choose a different color schema that then can also be re-used in subsequent layers.

Color Range

In gauge charts, you can customize the appearance of the gauge by applying colors to certain value ranges.

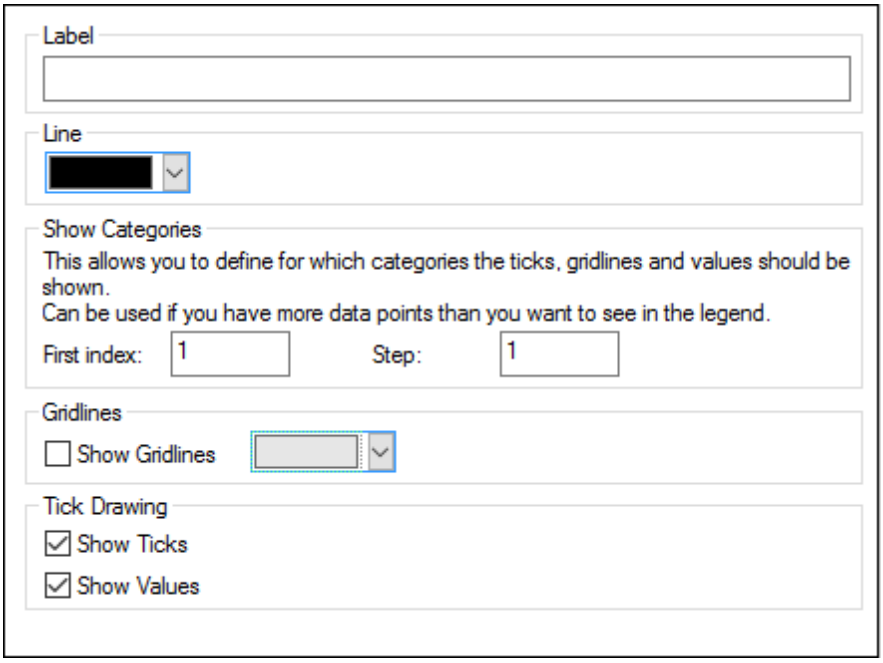
The definition shown in the screenshot above will appear in the gauge charts as follows:



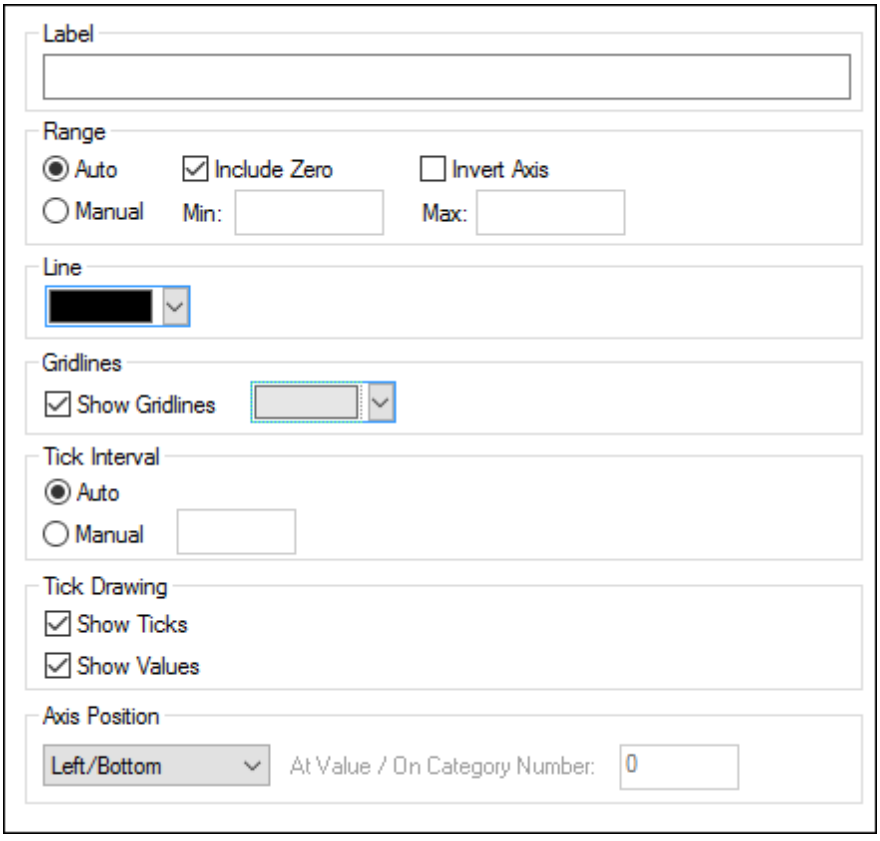


X-Axis

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`.



In Value Line Charts however, you can also define the value range, and define at what interval tick marks should be displayed.



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 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.

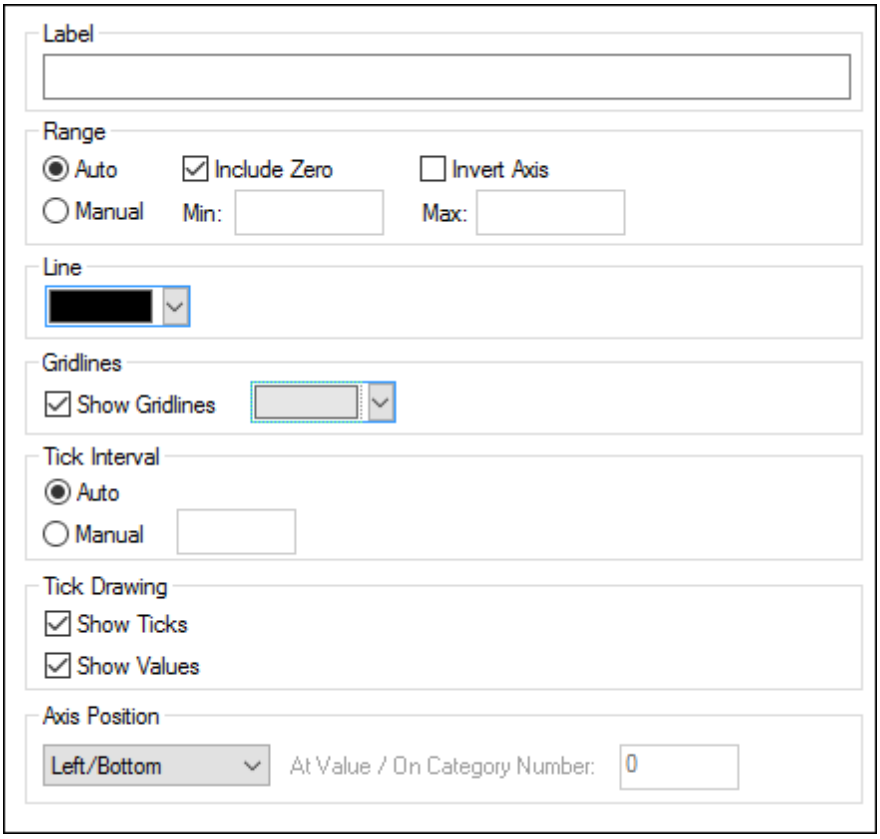
Tick Interval
If you are not satisfied with the default tick marks, you can activate the Manual radio button in the Tick Interval group box and enter the difference between the individual tick marks into the corresponding field.

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.

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
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 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 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.

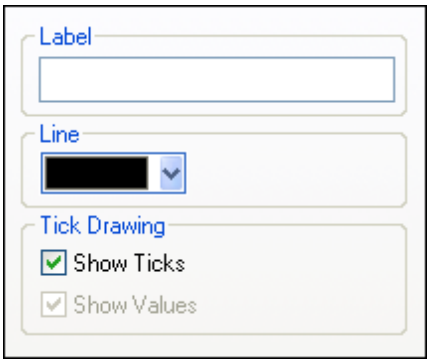
Tick Interval
If you are not satisfied with the default tick marks, you can activate the `Manual` radio button in the Tick Interval group box and enter the difference between the individual tick marks into the corresponding field.

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.

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).



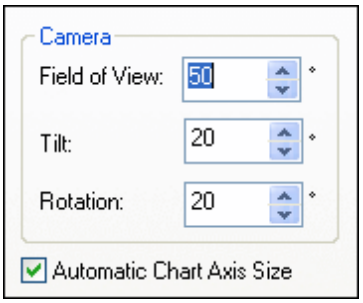
Label
The text entered into the `Label` field will be printed to the right of the axis as a description of the Z-axis.

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.

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.



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 activating the corresponding check box.

If the `Automatic Chart Axis Size` check box is activated, 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.

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.

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

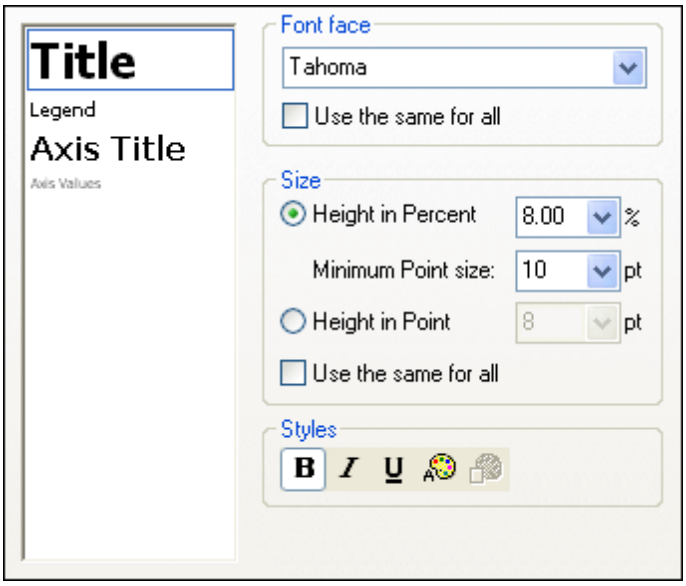
Needle Length	In round gauge charts, the length of the needle. (Note that the percentage is calculated from the diameter of the gauge, so if you choose a value greater that 50%, the needle will point to somewhere outside the gauge!)
Needle Width at Base	In round gauge charts, the width of the needle 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.
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!)

Fonts

The Fonts section of the **Change Appearance** dialog box lets you configure fonts for objects in the Chart window.



Font settings

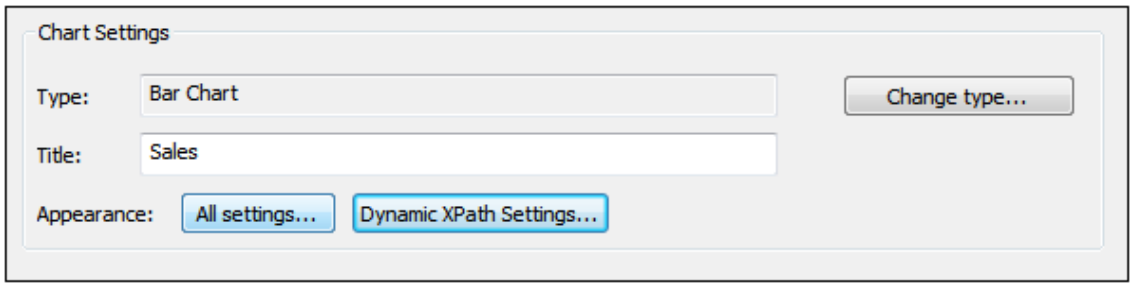
You can choose the font face, size, and style for the individual elements displayed in the Chart window. You can define the size as a percentage of the chart size and define a minimum size in points, or specify an absolute value (in points). To apply the same font and/or size to all text elements, activate 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 pies 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

Dynamic XPath Settings

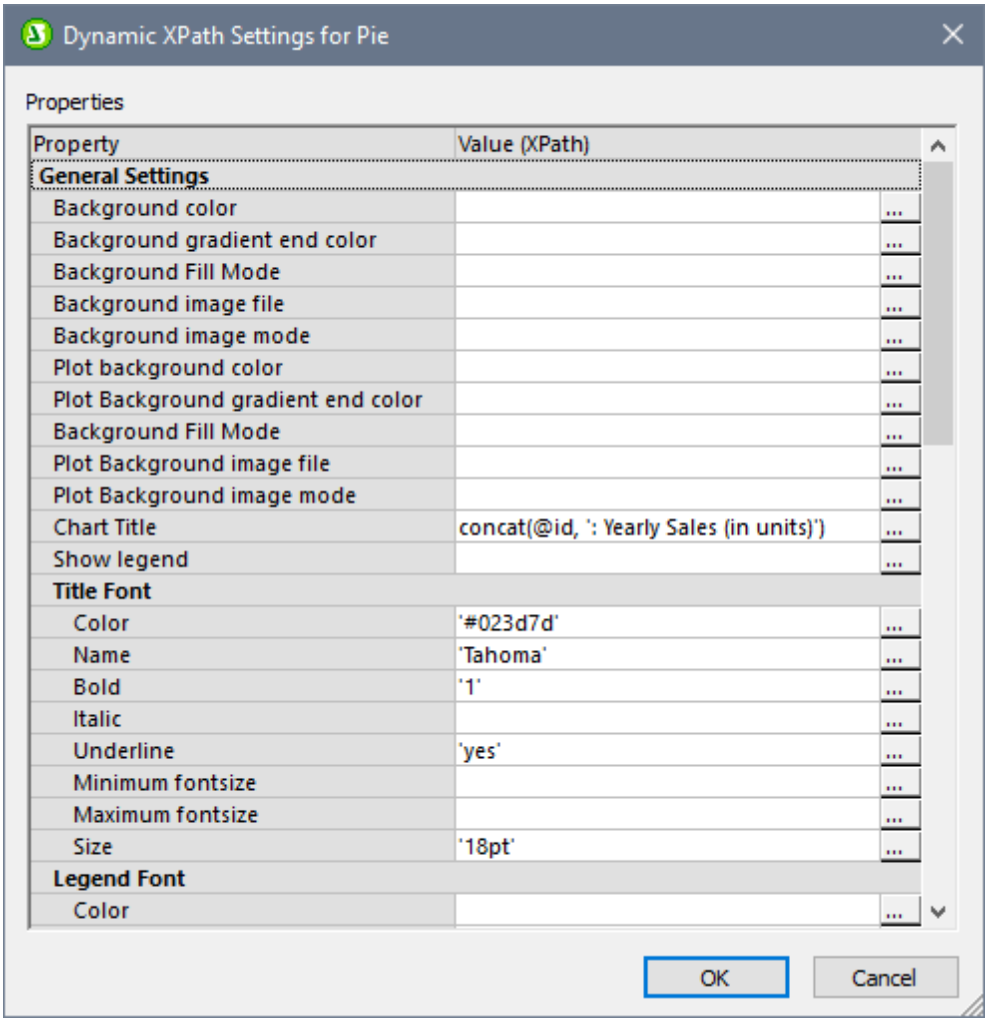
Dynamic XPath Settings are very useful if you wish to use dynamic data from the XML document in the settings of the chart. For example, the title of a chart about a `Region` element might need to have data about that `Region` element (such as its name) in the title of the chart. If there are several 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 of the Chart Configuration dialog (*see screenshot below*).



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.

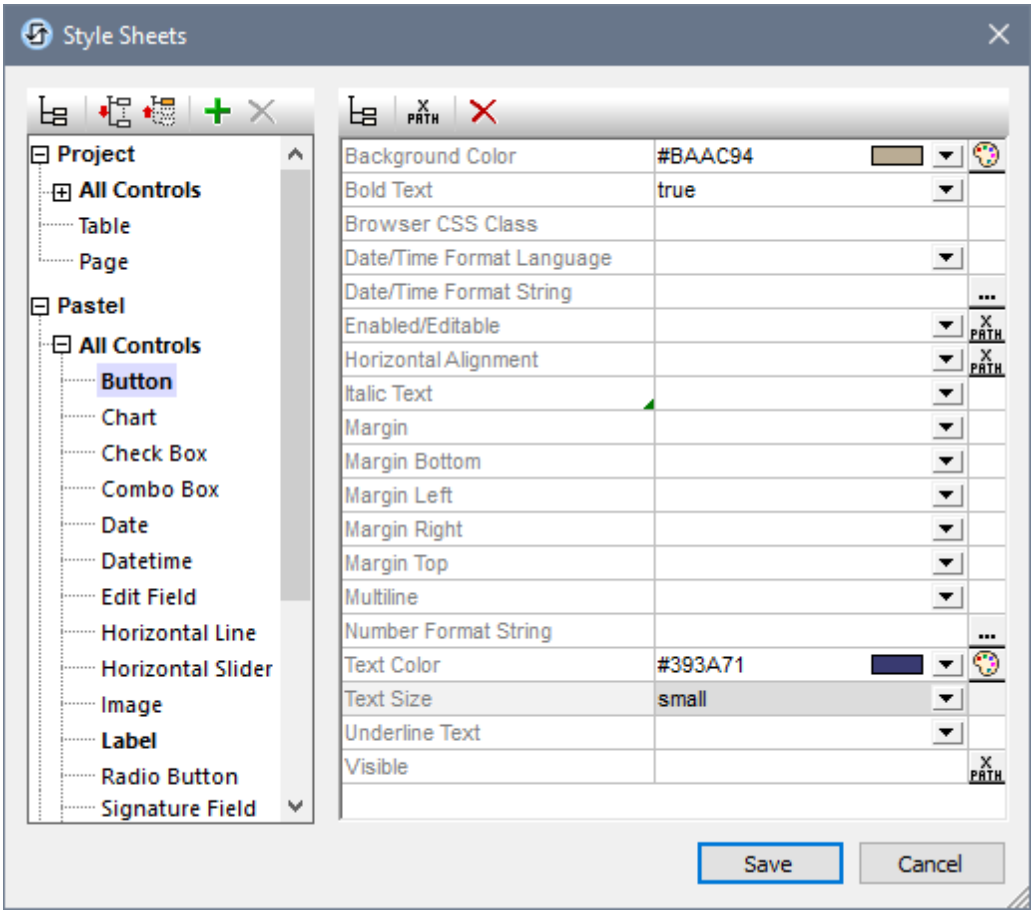


Note the following points:

- 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. 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 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'`.

11.7 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.



Adding 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 can be applied separately to individual pages, tables, and controls.

To add a user-created style sheet, click **Add Style Sheet** . To rename a user-created style sheet, double-click the name and edit.

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*) 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, 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 itself provide you with additional flexibility for defining cascading effects.

A user-created style sheet can be applied 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. 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.

This section

This section is organized as follows:

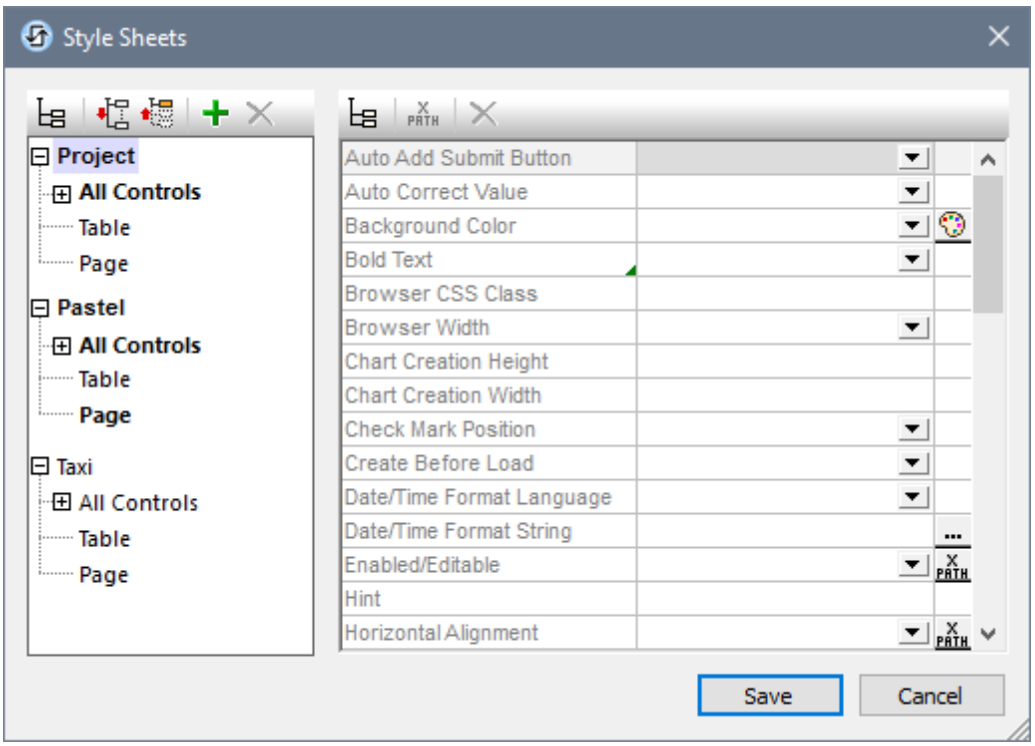
- Style Sheet Type and Scope describes the two types of style sheet and their respective scopes
- Priority within a Style Sheet describes the priority levels available within a single style sheet
- Priority across Style Sheets describes how styles can be prioritized when multiple style

sheets are used

- Applying User-Created Style Sheets shows how user-created style sheets can be applied to design components
- Style Sheet Properties provides an overview of how to work with component styles in the Style Sheets dialog

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

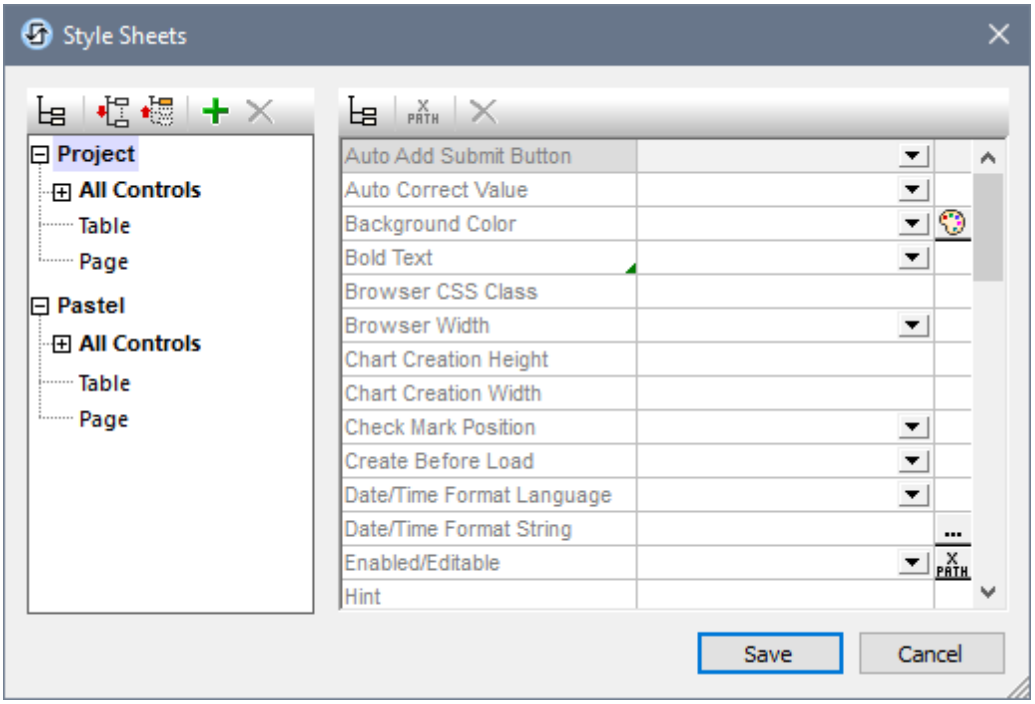
list of only the styles that have been defined; for example, when you want an overview of currently defined styles. The left-hand pane also has toolbar icons for (i) expanding all items, and (ii) collapsing all items.

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)
```

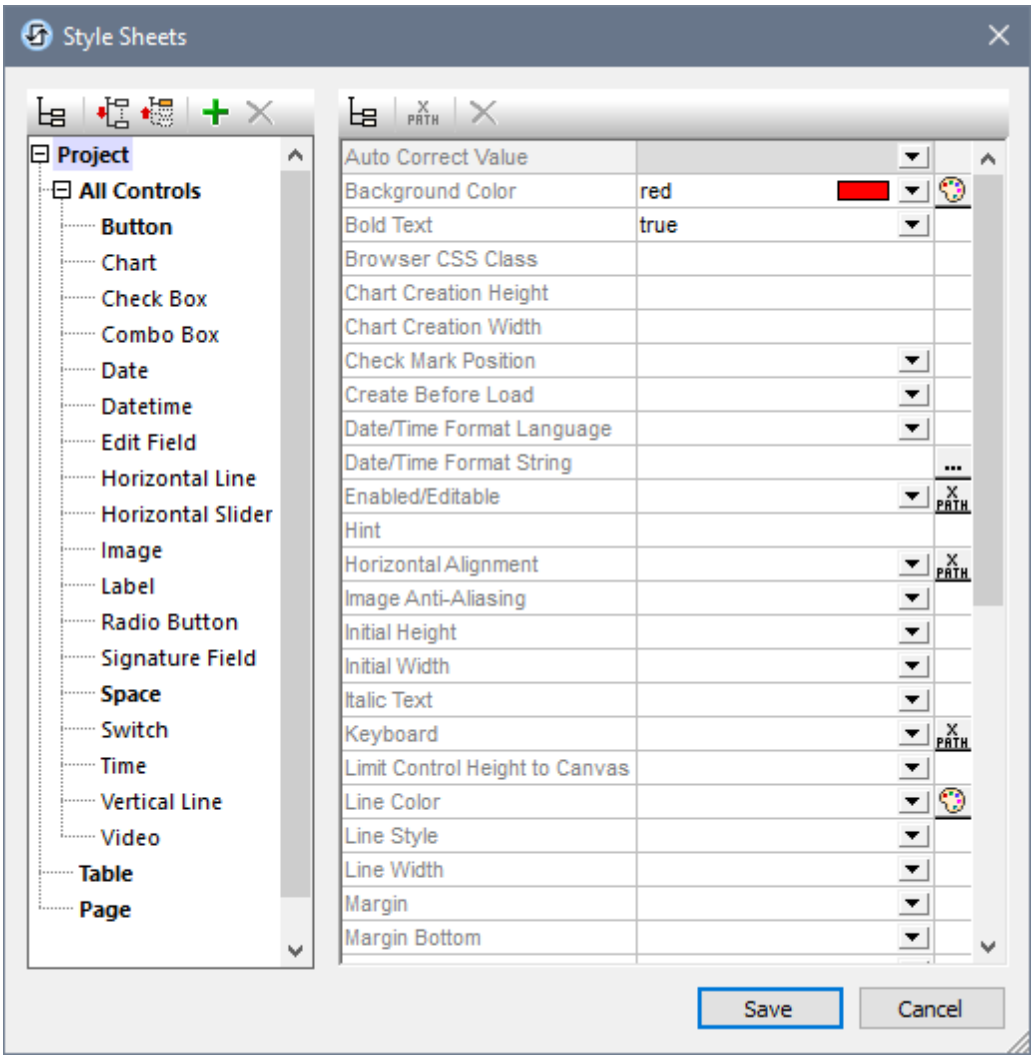
You can also see this hierarchy in the screenshots below.

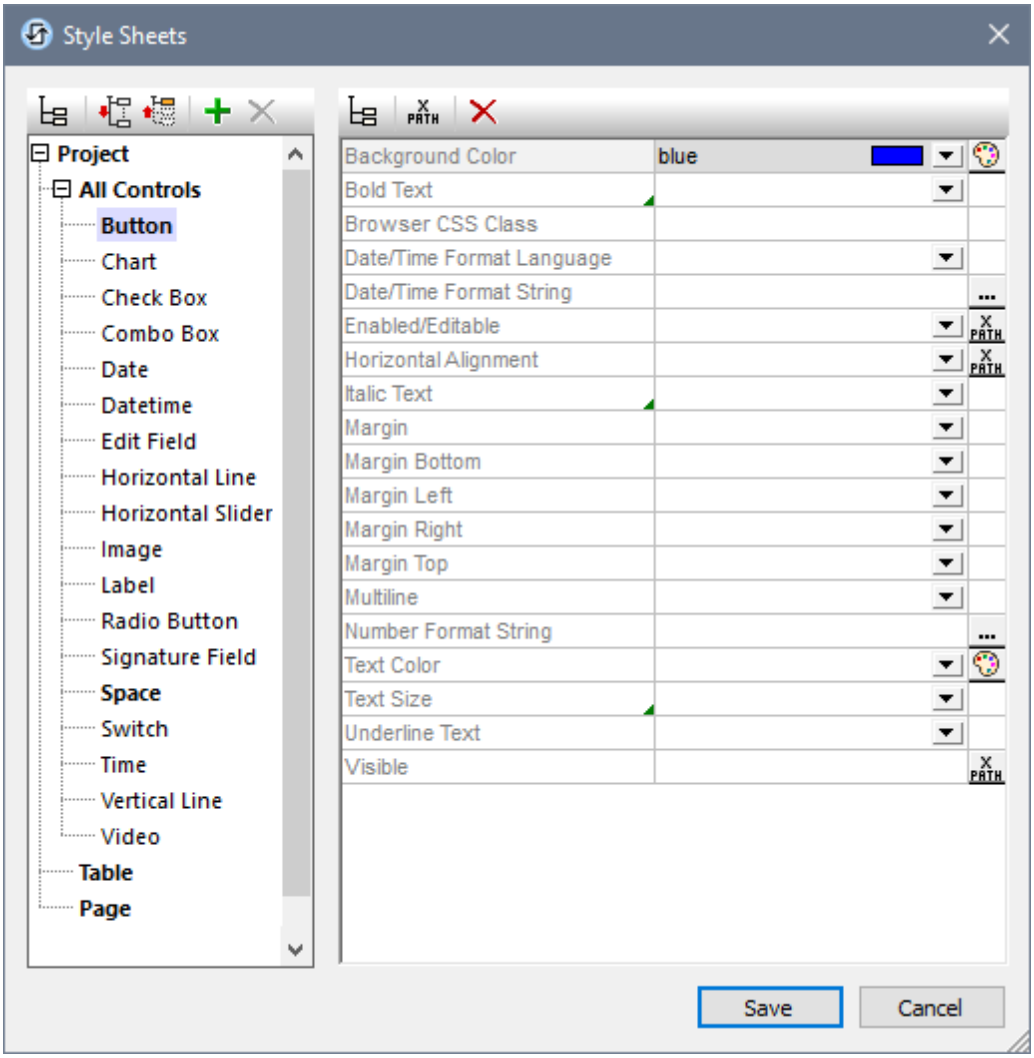


Each higher level in the hierarchy passes all its style properties to the level below it. So the *All Controls* level passes all its style definitions to the control types on Level 3. The principle is that, if a property value is set at the *All Controls* level, then all the controls (on the lower level) 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.)





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	Style sheet property of table	Style sheet property of page

set on...	set on...	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. 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	--	--

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**.

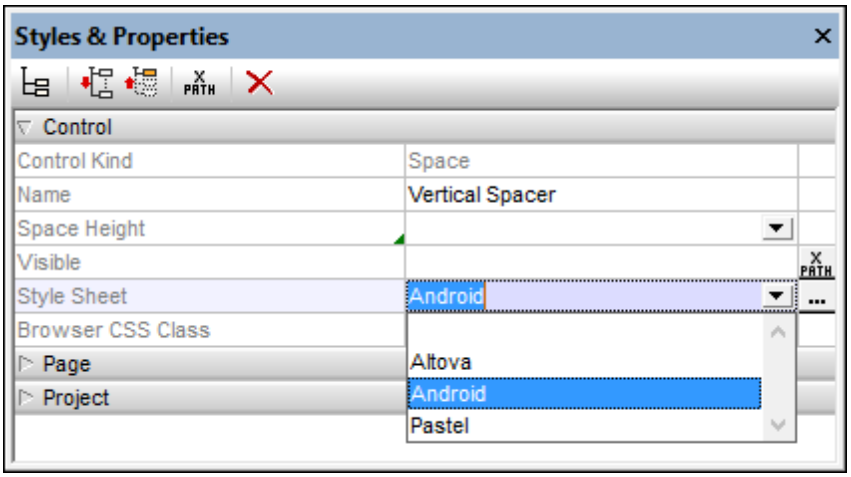
Like any other style definition, platform default values can override values that are defined relatively farther away (from the component), and can be overridden by style definitions that are relatively closer to the component.

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.



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.

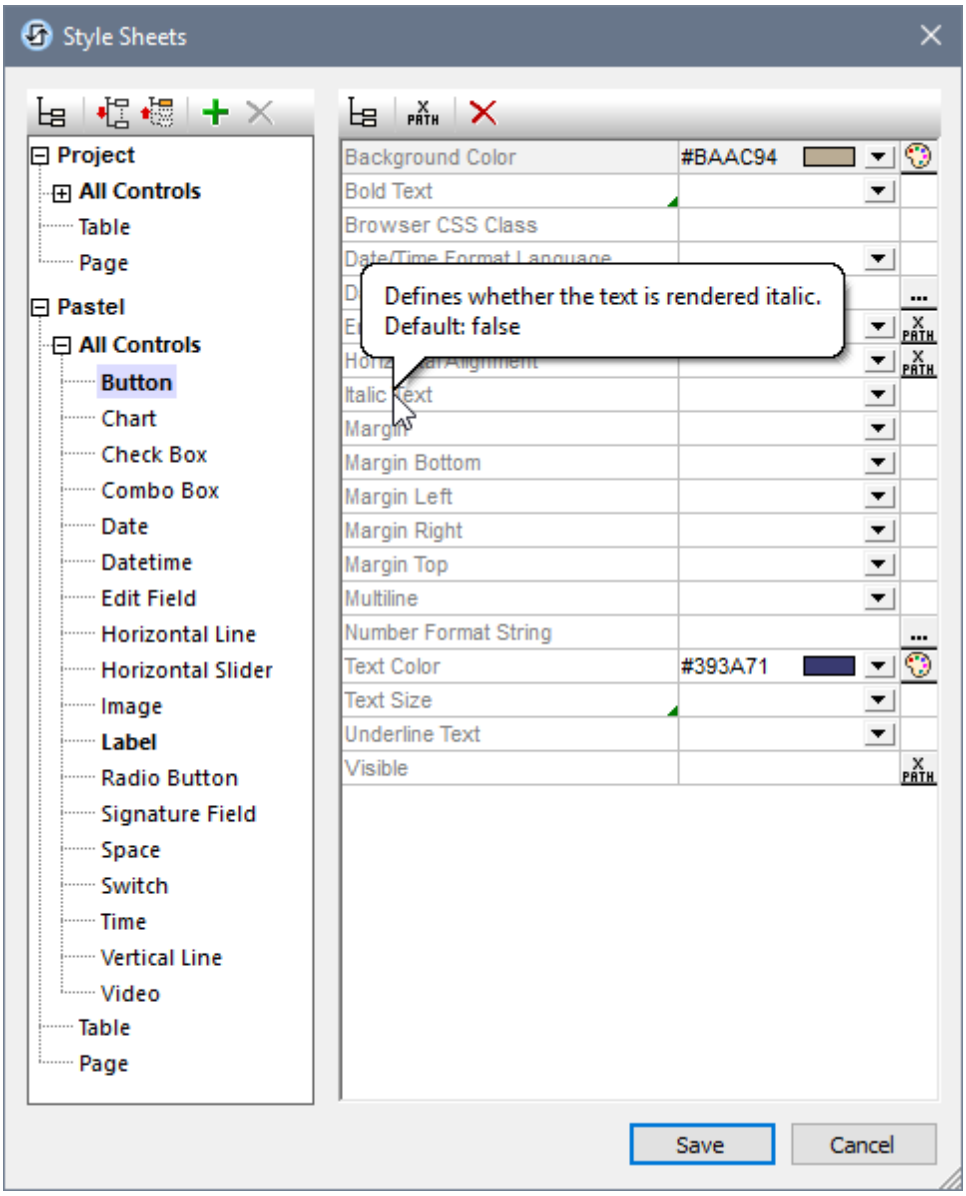
Advantages of a style sheet selection via XPath

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.

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.



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 of that property. Do this by right-clicking the property and selecting **Set Platform Default Value**.

11.8 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 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.
- In the design's Rich Text Style Sheets dialog, you can define style rules for different elements and style mappings for existing style rules.

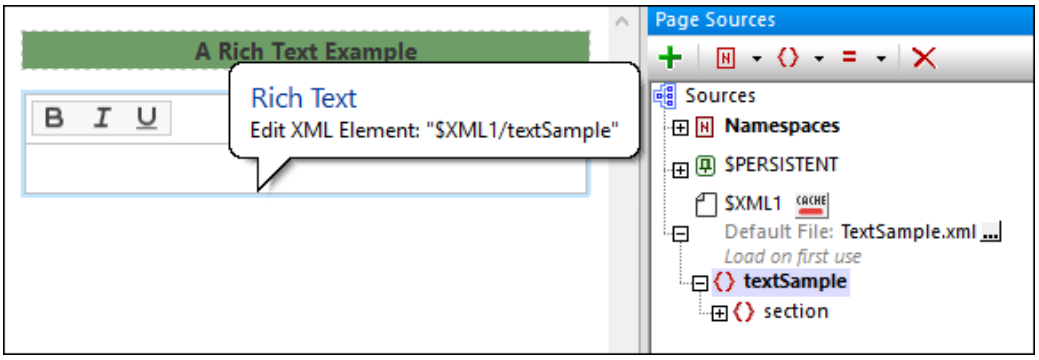
In this section
This section is organized into the following sub-sections:

- The Rich Text Control
- Rich Text Style Sheets: Setup
- Rich Text Style Sheets: Styles
- Editing Rich Text Content

The Rich Text Control

The first step towards enabling Rich Text in your design is to add the Rich Text control 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 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 via the **Project | Rich Text Style Sheets** menu command, and create a new style sheet. You can define multiple style sheets for the project. This enables you to assign different style sheets to different Rich Text controls. How to set up a style sheet is described in the section Rich Text Style Sheets.
4. After you have created the style sheet, select the Rich Text control in the design and, via its `Rich Text Style Sheet` property, assign the style sheet to the control. Note that each Rich Text control 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`** property.

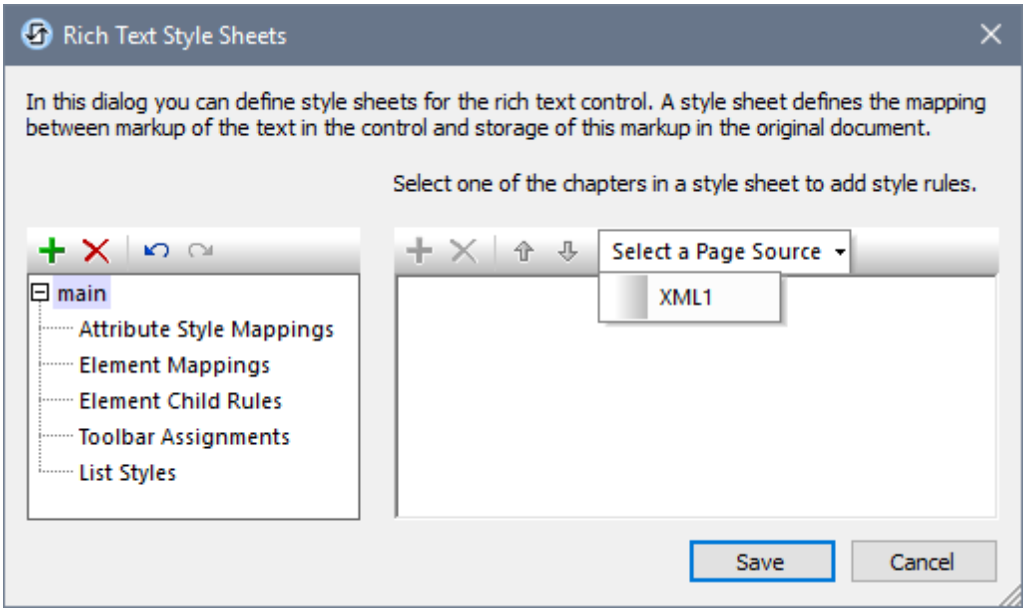
Rich Text Style Sheets: Setup

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). You can define multiple style sheets for a project. Any one of these style sheets can be assigned to a Rich Text control. The styles in that style sheet are then used for the styling of the text displayed in that Rich Text control.

Create a style sheet

To create a Rich Text style sheet, do the following:

1. Select the menu command **Project | Rich Text Style Sheets**. The Rich Text Style Sheets dialog (*see screenshot below*) appears.
2. Click **Add Style Sheet** in the toolbar of the left pane.
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.

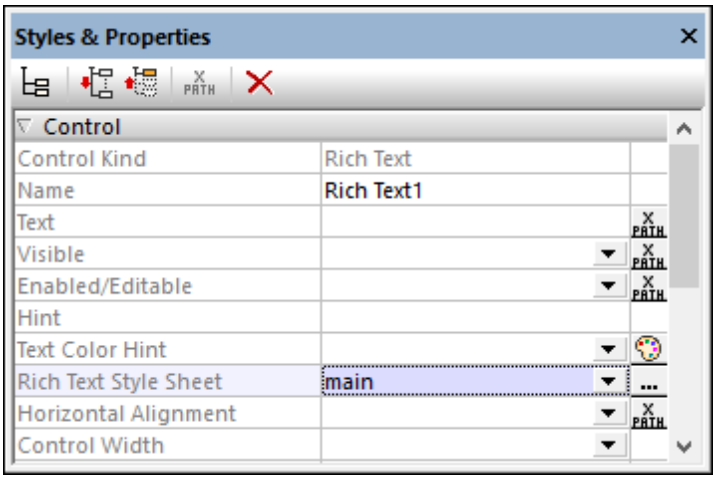


How to create style mappings and rules is described in the section Rich Text Style Sheets: Styles.

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 via the control's `Rich Text Style Sheet` property. In the screenshot below, for

example, the Rich Text control has been assigned the style sheet named **main**.



Rich Text Style Sheets: Styles

After a Rich Text style sheet has been created, 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.

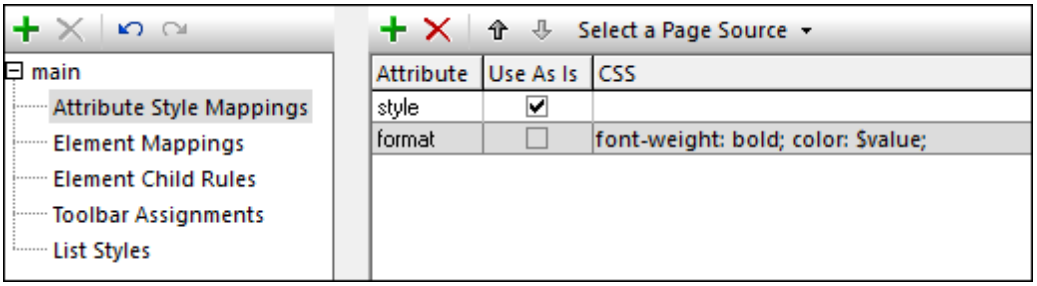
In the Rich Text Style Sheets dialog, the style rules and mappings are organized into sections. Each of these sections is described below:

- Attribute Style Mappings
- Element Mappings
- Element Child Rules
- Toolbar Assignments
- List Styles

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. 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.



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

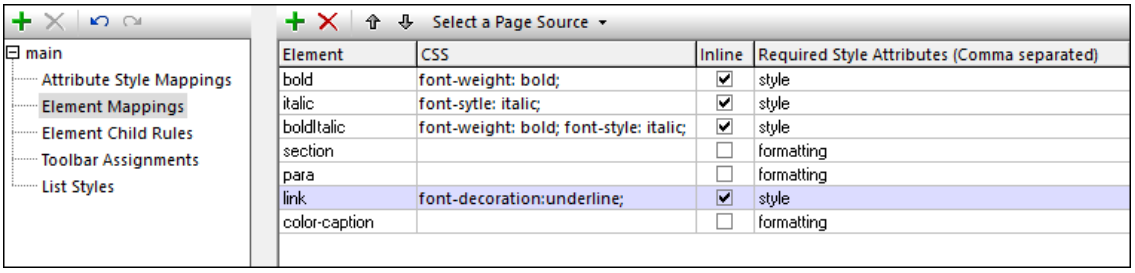
- If the attribute definitions shown in the screenshot above are applied to the following element in the XML of the page source:
  <heading style="font-style:italic;" global="font-weight:bold;">Text Formatting**</heading>**
  then the resulting style definition will be **font-style:italic;** since the **@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* 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. 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.

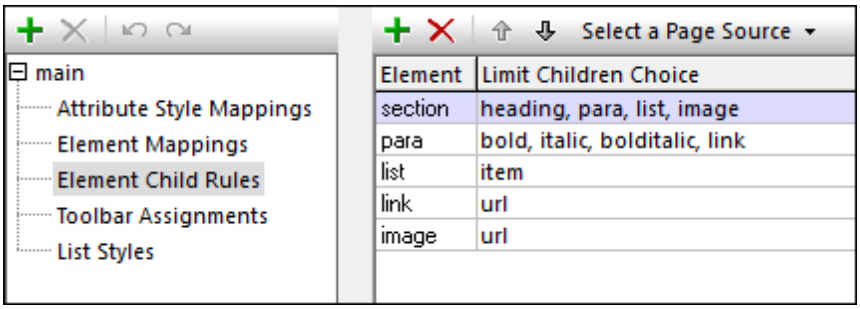


You can add an element to the list by clicking **Add Style** in the toolbar of the right-hand pane and entering the name of the element you want to define.

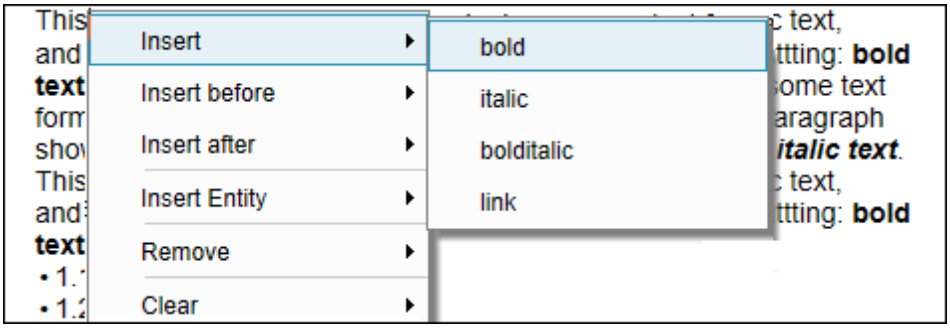
Element Child Rules

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. 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.



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.

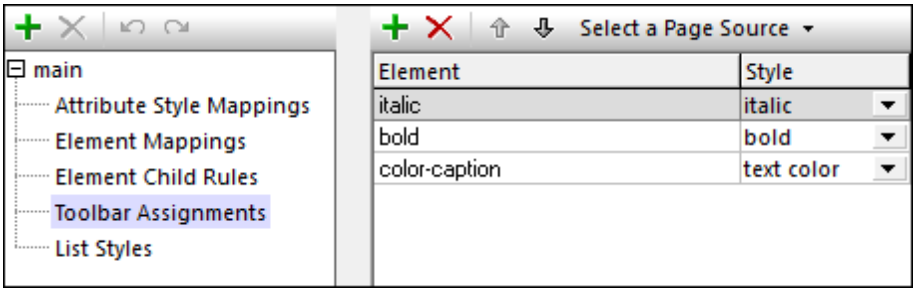


Toolbar Assignments

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 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 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 section (see the definition of **`color-caption`** in the screenshot there).



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 are **not displayed**. In the designer (during simulations), 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.

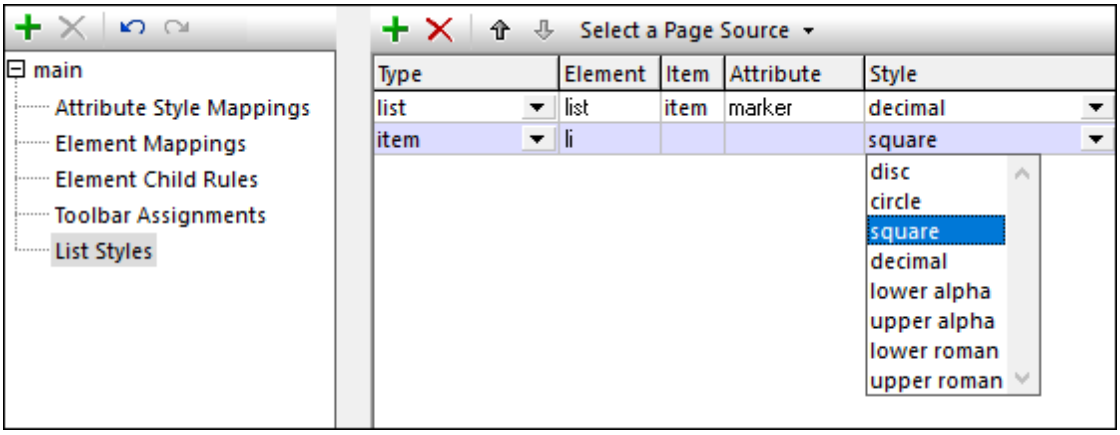
List Styles

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*).

Note: The selected marker determines whether the list is a numbered list or itemized list.



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.

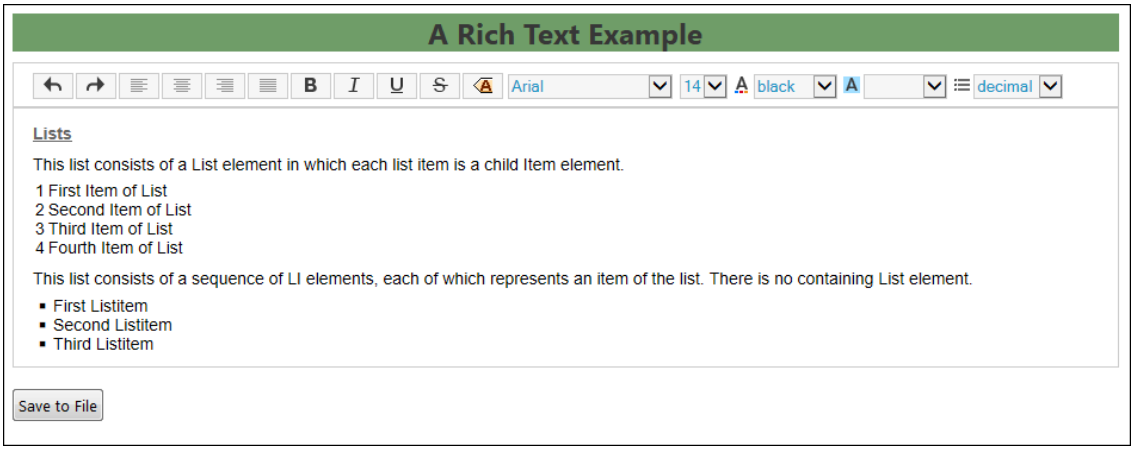
Editing Rich Text Content

Text content in a Rich Text control 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`** property.



The control's toolbar

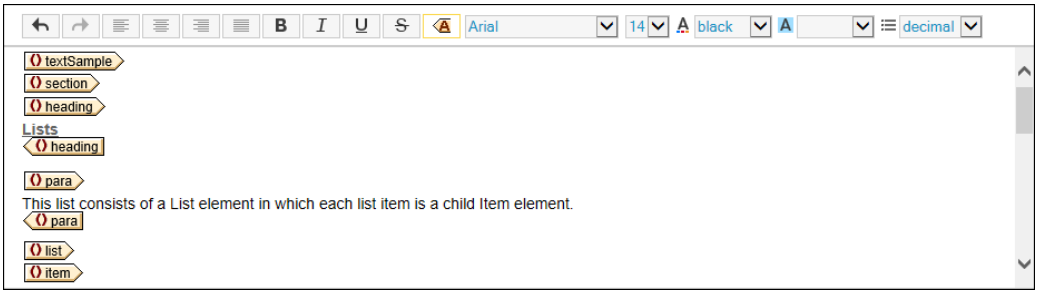
The toolbar of the Rich Text control (*screenshot below*) is displayed on Web and Windows clients only.

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.
- 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.)

- 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 to toggle element markup tags on and off (*the screenshot below shows markup tags switched on*).



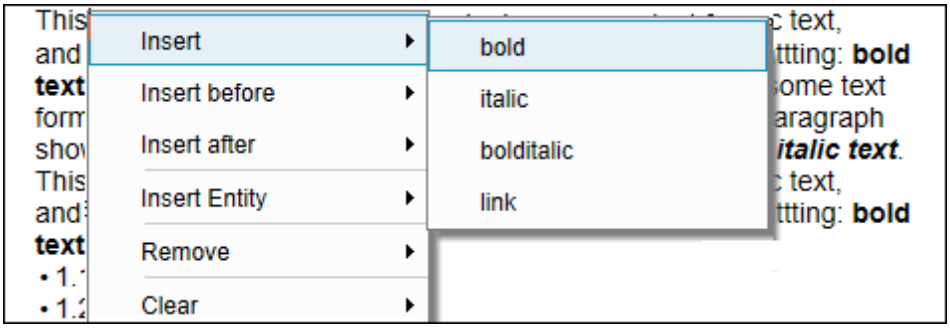
- Four combo boxes in which to select, respectively, the font family, font size, font color, and font background.
- A combo box in which to select the list-item marker of (i) the entire list (in the case of Container lists), or (ii) the selected list-item (in the case of Item-sequence lists).

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 are **not displayed**. In the designer (during simulations), 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 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*).

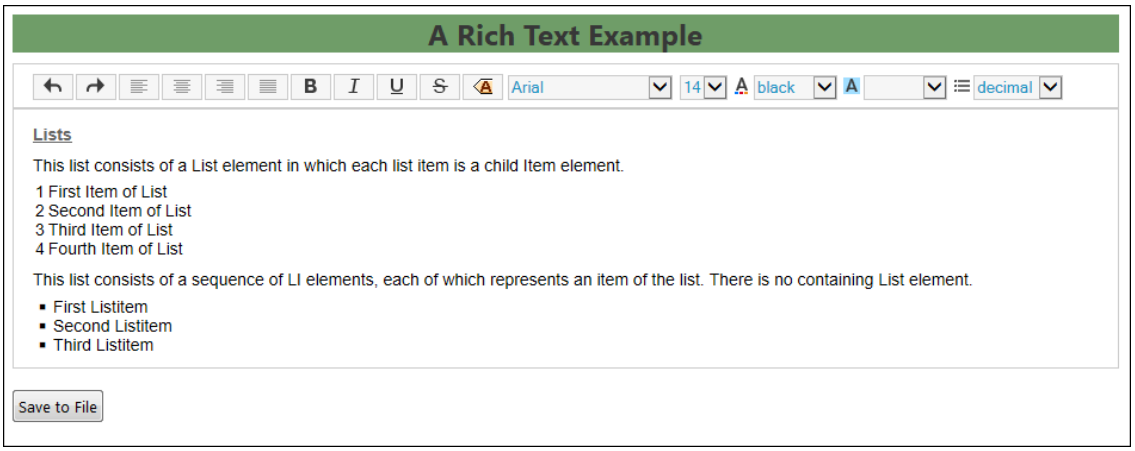


The elements that are shown in the context menu depend on the element in which the cursor is placed:

- *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 while the second list is an Item-sequence list.



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 is selected, then the markers of all items in the list are changed. If an item in an Item-sequence list 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.

11.9 Hyperlinking to Solutions

You can create hyperlinks to solutions in the following ways:

- Via the Open URL action of page or control events
- In an email that the end-user sends

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

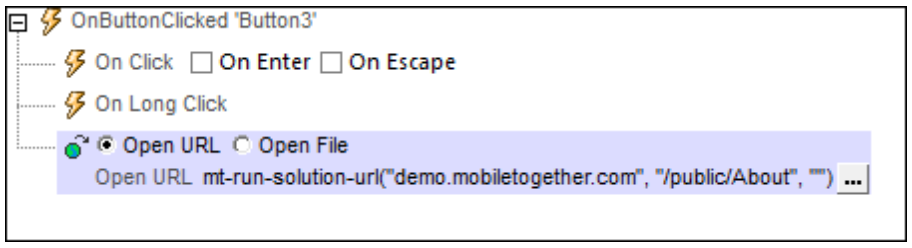
Note: Links to update server settings do not work in Gmail and some other email applications, but they work in popular clients such as AquaMail, K9, and MailWise. They have been tested in AquaMail and K9 and work correctly in these applications.

Linking to a solution from a design component

A design component can be linked to a solution via the component's Open URL action. For example, if a button is clicked, the button's Open URL 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 action (*see screenshot below*).
2. Create an XPath expression that uses the **mt-run-solution-url** function to generate the URL of the solution. The function is described below.



▼ mt-run-solution-url

mt-run-solution-url(ServerAddress? *as xs:string*, **SolutionName?** *as xs:string*, **InputParameters?** *as xs:string*) as **xs:string?**

Generates the URL of a solution from the three submitted arguments:

- `ServerAddress`: Takes the name or IP address of the MobileTogether Server on

which the solution that you want to run is deployed
- `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)
- `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 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`** 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 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-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:

- If the first argument, `ServerAddress`, is the empty string, then the current server is used.
- The first `ServerAddress` argument is used to look up server information stored on the client. 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.
- If the second argument, `SolutionName`, is the empty string, then the current solution is used.

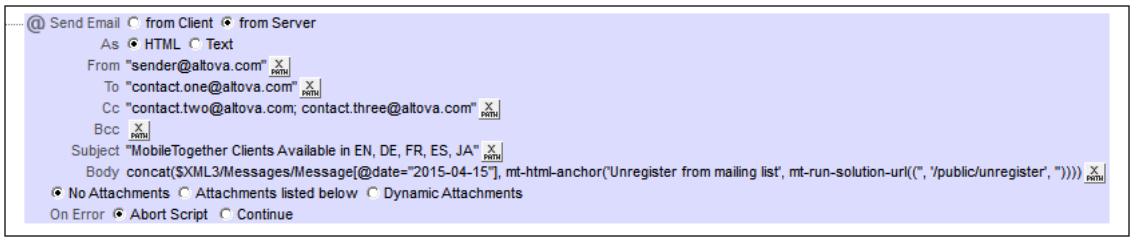
- The second argument, `SolutionName`: (i) generates the deployed path (on the server) if the solution is run on the server, but (ii) generates a file path for simulations.
- The third argument, `InputParameters`, uses another <%MT%>-specific XPath extension function called **`mt-run-solution-url-parameters`** 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 as a sequence of string values in the **`$MT_InputParameters`** global variable of the target solution. The order of the string values in the **`$MT_InputParameters`** sequence is the same as that in the sequence submitted to the **`mt-run-solution-url-parameters`** function for generating the URL's query parameters. Since the order of string values in the **`$MT_InputParameters`** is therefore known to you, each string can be accessed In XPath expressions by using position predicates. For example: **`$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 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`** function in the XPath expression of the *Body* option (*see screenshot below*).



The **`mt-html-anchor`** 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** 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` 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` 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.

Chapter 12

XPath/XQuery: Expressions, Functions, Variables

12 XPath/XQuery: Expressions, Functions, Variables

This section is organized into the following sections:

- XPath/XQuery Expressions and Functions
- Global Variables

For a description of functions in Altova's general XPath extension function library, see the section Altova Extension Functions. (These extension functions work with all Altova products, including MobileTogether.)

12.1 XPath/XQuery Expressions and Functions

This section groups together the XPath/XQuery functionality of MobileTogether Designer: the Edit XPath/XQueryExpression dialog, and the extension functions and user-defined functions that MobileTogether Designer makes available to each project.

- Edit XPath/XQuery Expression Dialog
- MobileTogether Extension Functions
- User-Defined XPath/XQuery Functions
- FAQ about XPath/XQuery

Edit XPath/XQuery Expression Dialog

The **Edit XPath Expression** dialog (*screenshot below*) is used to create and edit XPath expressions for a range of MobileTogether features. For example, file paths in many dialogs can be composed with XPath expressions; this allows the dynamic composition of file paths, and enables paths to be based on node content in any page source.

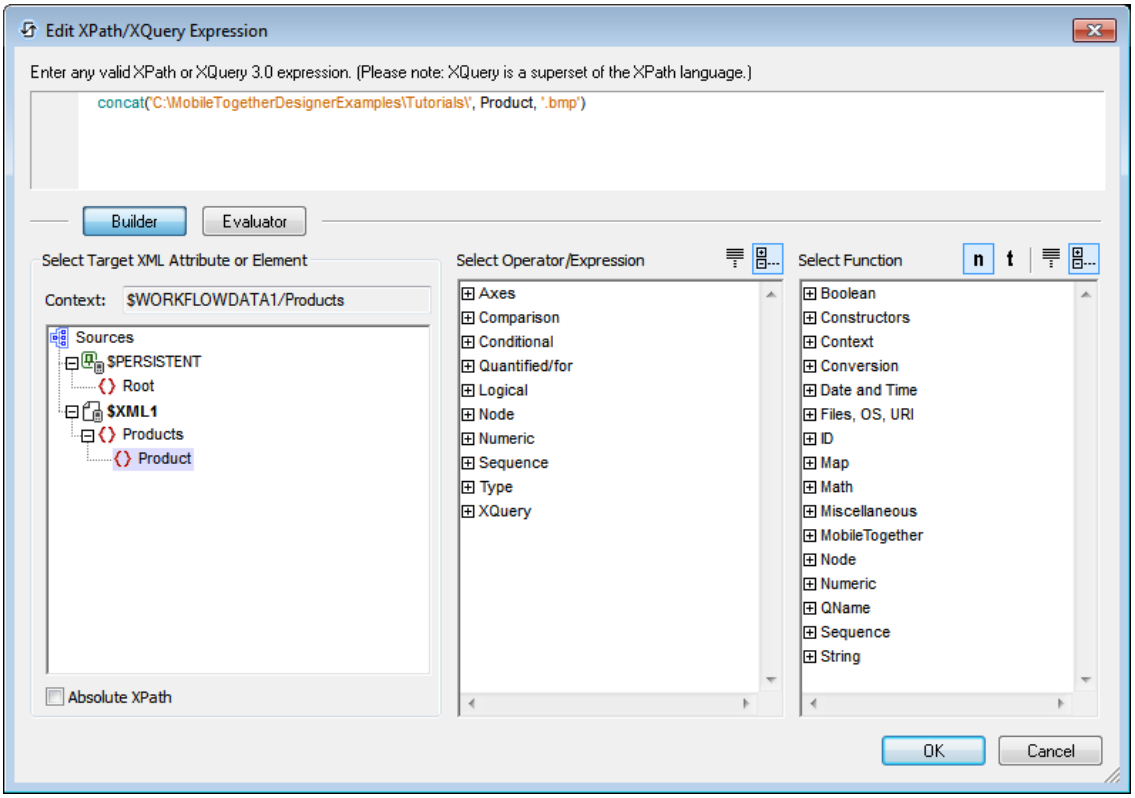
The Edit XPath Expression dialog provides a tree of XML data sources and a library of XPath/XQuery 3.1 operators and functions (*see screenshot below*), and thus supports the building of valid XPath/XQuery 3.1 expressions. The dialog has two modes: (i) Builder mode, for creating XPath expressions, and (ii) Evaluator mode for checking the result of the XPath expression being currently edited. You can switch between the two modes by clicking the respective buttons (**Builder** and **Evaluator**).

Click **OK** when you have completed editing the XPath expression.

Note: Text that is copied from the Edit XPath Expression dialog can be pasted as XPath into the Styles & Properties Pane.

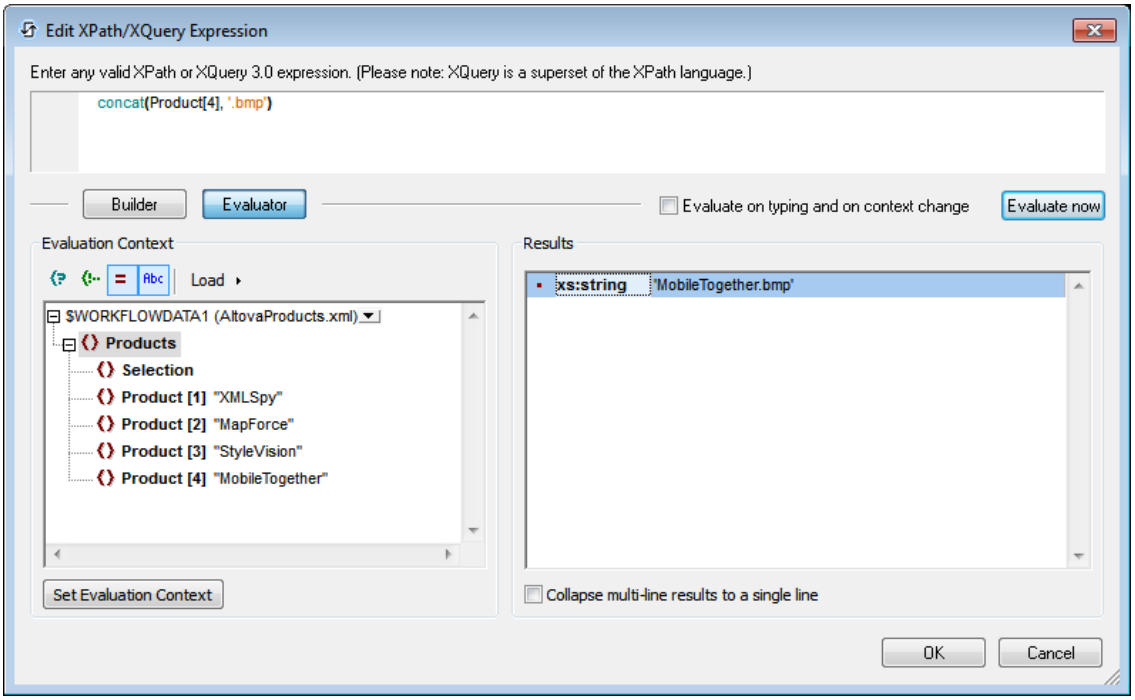
Builder mode

In Builder mode, you can build XPath expressions quickly and correctly by either (i) entering the XPath expression in the Expression text box via the keyboard, or (ii) using the entry helpers of Builder mode to insert nodes, operators, and functions by double-clicking them in their respective lists (*see screenshot below*). When an expression that has been entered in the Expression text box contains errors, the expression is underlined in red, thus alerting you to the problem. Builder mode is described in detail in the section, XPath Expression Builder.



Evaluator mode

In Evaluator mode, you can see, in the *Results* pane on the right-hand side of the dialog (*see screenshot below*), the results of evaluating the currently entered XPath expression. The *Evaluation Context* pane shows the structure and contents of the currently assigned Working XML document. The Edit XPath Expression dialog's Evaluator mode is described in detail in the section, XPath Expression Evaluator.

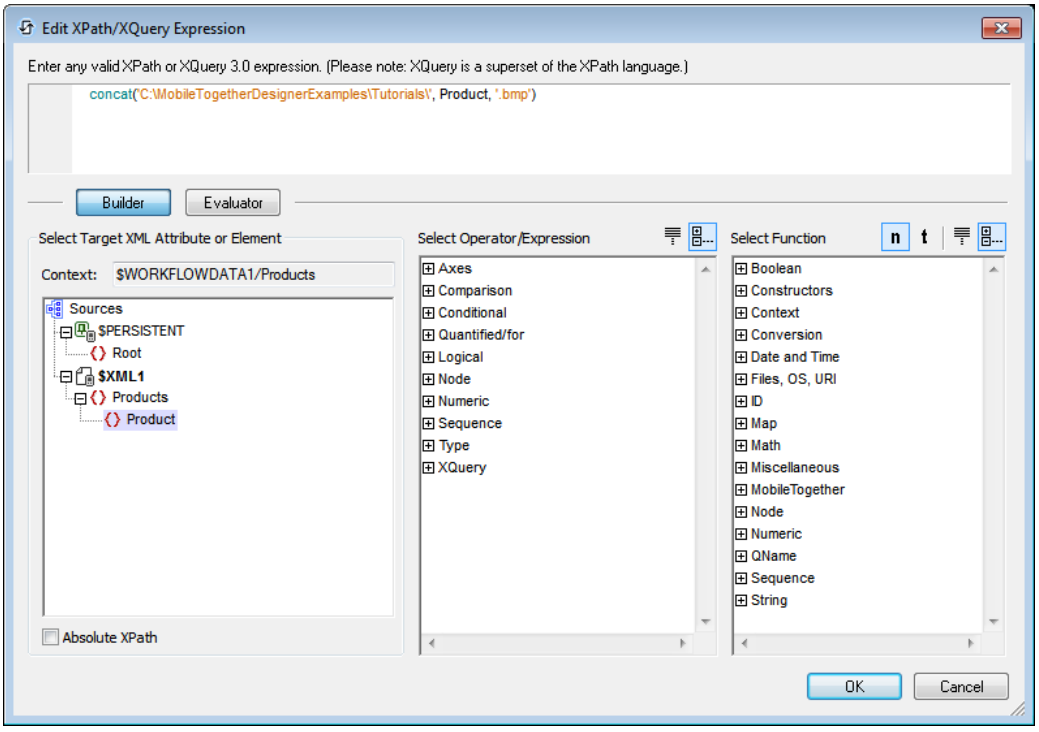


XPath/XQuery Expression Builder

When the **Builder** button in the Edit XPath Expression dialog is clicked (*see screenshot below*), entry helper panes to help you build an XPath expression become visible. Double-click an entry in any of these entry helpers to enter it at the current cursor point in the XPath expression.

There are three entry helper panes:

- A schema tree for entering element and attribute nodes in the XPath expression. If the *Absolute XPath* check box is unchecked, then the location path to the selected node is entered relative to the context node (the node in the design within which the XPath expression is being built). An absolute XPath expression starts at the document root. Absolute paths are used for the selected node if the *Absolute XPath* check box is checked.
- An entry helper pane for: (i) axes (`ancestor::`, `parent::`, etc), (ii) operators (for example `eq` and `div`), and (iii) expressions (`for # in # return #`, etc). This pane displays the axes, operators, and expressions either listed alphabetically or grouped by functional category. Select the option you want by clicking the appropriate icon above the pane.
- An entry helper with the functions of the active XPath version either listed alphabetically or grouped by functional category. Select the option you want by clicking the appropriate icon above the pane. The **n** and **t** buttons above the pane display the arguments of functions, respectively as names and datatypes.



Building XPath expressions

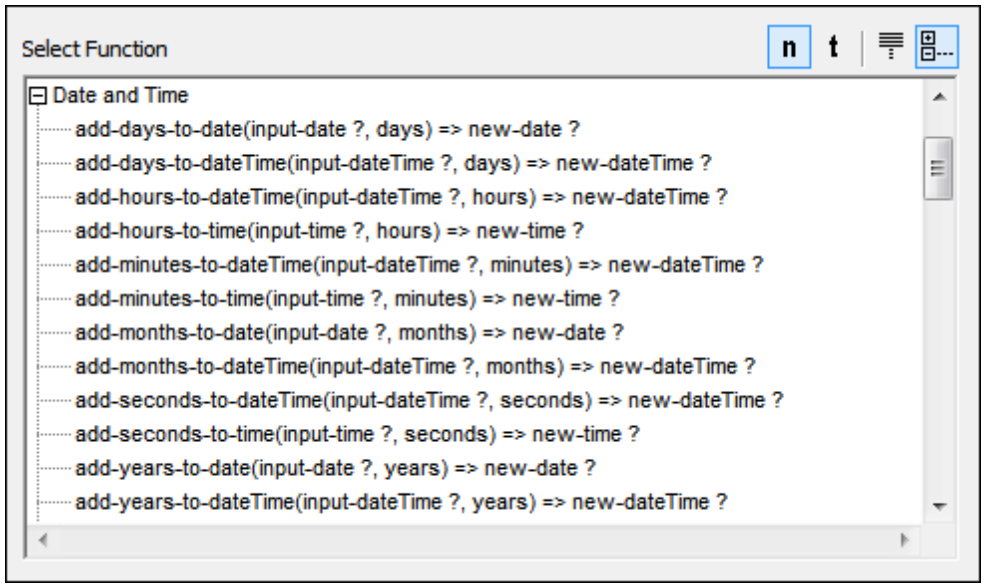
The Edit XPath Expression dialog helps you to build XPath expressions in the following ways.

- *Context node and schema tree*
  The *Context* text box in the *Select Target XML Attribute or Element* pane immediately shows you the context node. In the schema tree below the *Context* text box, 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 target XML tree*
  In the *Select Target XML Attribute or Element* pane, the structure of the target XML document is displayed. Double-click a node in the schema tree to insert that node in the XPath expression. If the *Absolute XPath* check box is not checked, the selected node will be inserted with a location path expression that is relative to the context node. For example, in the screenshot above, the `Product` element, which is a child of the `Products` element (the context node), has been inserted with a location path that is relative to the context node (that is, as `Product`). If the Absolute XPath check box were checked, the `Product` node would be inserted as `$XML1/Products/Product`.

- *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 (click the respective icon 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 (**?** 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 **n** or **t** button above 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` item. Placing the mouse cursor over a function displays a brief description of the function.



To insert a function in the XPath expression, double-click the required function.

Note: The XPath default namespace is used for all XPath/XQuery functions, including extension functions and user-defined functions.

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*).

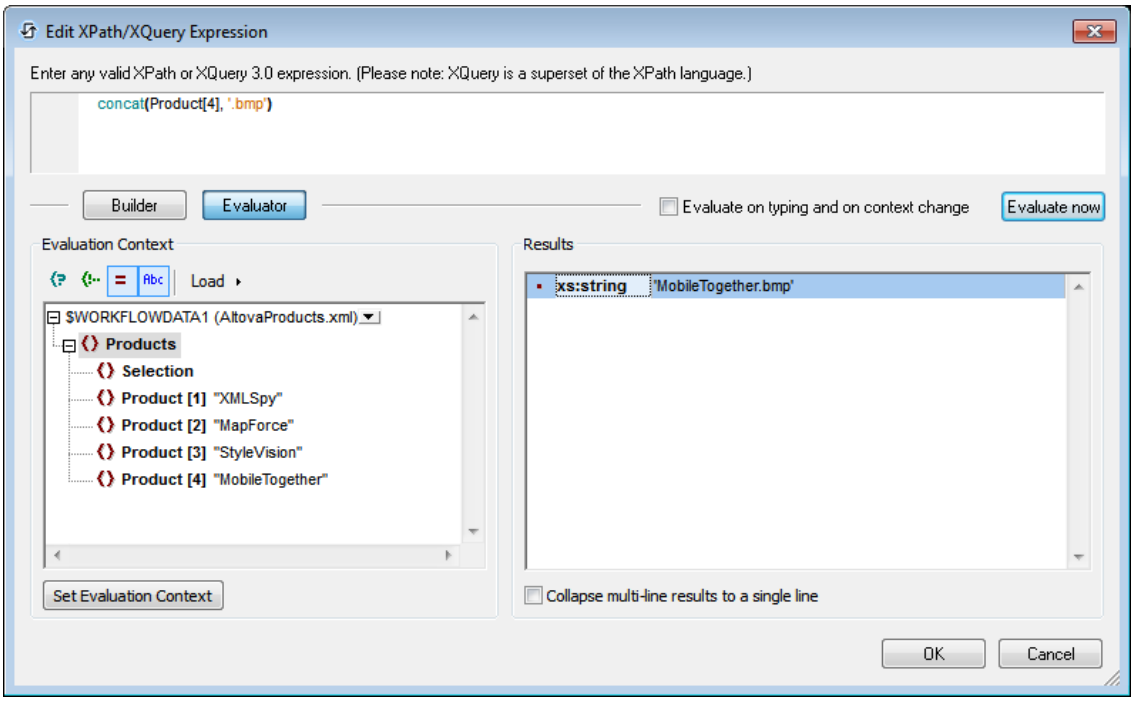
These include the following components:

- elements (such as `presswatch` in the screenshot above),
- descendant nodes (`presswatch/selection` in the screenshot above),
- XPath functions (`fn:upper-case` above) and XPath axes (`ancestor-or-self` above)
- a list of the global variables defined for the project (displayed when a **$** character is entered in the expression)
- a list of the custom strings defined in the Localization dialog (displayed when the `mt-load-string` function is entered in the expression; see the description of `mt-load-string`)

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.

XPath/XQuery Expression Evaluator

Clicking the **Evaluator** button in the Edit XPath Expression dialog switches the dialog to Evaluator mode (*see screenshot below*). The dialog in this mode has two panes: the *Evaluation Context* pane and the *Results* pane.



The XPath expression and its evaluation

The XPath expression in the *XPath Expression* text box can be edited, and the expression can be evaluated. The results of the evaluation are displayed in the *Results* pane. In the screenshot above, for example, the result of evaluating the XPath expression `concat(Product[4], '.bmp')` is displayed as the string `MobileTogether.bmp` (because `MobileTogether` is the content of the fourth `Product` element).

Note: The XPath default namespace is used for all XPath/XQuery functions, including extension functions and user-defined functions.

Using Builder mode and switching to Evaluator mode for the results

If you wish to use entry helpers to build the XPath expression, you can switch to Builder mode (by clicking the **Builder** button), build the expression in Builder mode, then switch to Evaluator mode to see the results of the evaluation.

When is the XPath expression evaluated?

Evaluation is carried out in two mutually exclusive situations:

- *Evaluate on typing:* If this check box is selected, the XPath expression is evaluated: (i) with every keystroke used to edit the expression, and (ii) when the mode is switched from Builder mode to Evaluator mode.
- *Evaluate now:* This button is enabled when the *Evaluate on Typing* option is not checked. Click it to evaluate the expression.

The Evaluation Context pane

The *Evaluation Context* pane shows the structure and contents of the data sources. Nodes in document trees can be expanded or collapsed by clicking the respective icons of individual nodes. You can load an XML file structure by clicking **Load** and browsing for the file you want.

The icons above the pane display or hide 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.

Changing the context node for evaluation purposes

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**. If the *Evaluate on Typing or on Context Change* option is checked, then the result will appear immediately in the Results pane.

This feature is useful for checking results with different context nodes. Note, however, that the actual context node for the expression will be the context node within which the current design component is being created. At runtime, the actual context node will be used, not the context node used in the Evaluator.

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 event in conjunction with the Update Display action.

Note: For a description of functions in Altova's general XPath extension function library, see the section Altova Extension Functions. (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-languages

Description
Returns the languages that have been defined in the Localization dialog.

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 hat 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 data source node that provides such text can be submitted.

Usage
`mt-base64-to-hexBinary``($XML1/Element1/@image)` converts a Base64 image to hexBinay

▼ 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-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**
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.

▣ *Examples*

- **mt-client-ip-address**`()` returns the client IP address; in simulations, returns the value set in the Options tab of MobileTogether Designer

▼ 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-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.

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. 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 or Button control) is used. Consequently only the `Control Kind` parameter is mandatory.

- `"Control Kind" : "Label"|"Button"`
- `"Text Size" : "small"|"medium"|"large"`
- `"Unit" : "px"|""` *default is* `"px"`
- `"Bold Text" : "true"|"false"`
- `"Italic Text" : "true"|"false"`
- `"Underline Text" : "true"|"false"`
- `"Button Look"` : Any of the `Button Look` options (for example, `"+"|"-"|">"|"Share"`)

▼ *Examples*

- `mt-control-width("Send", map{"Control Kind" : "Button", "Text Size" : "medium", "Unit" : "", "Bold Text" : true(), "Italic Text" : false(), "Underline Text" : false(), "Button Look" : "+"})`

▼ 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.

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 unmodified. The function tests whether there was any modification to the DB.

Usage
```
mt-db-any-changed-rows($DB1)
```

▼ mt-db-deleted-original-fields

Description
Returns field attributes from the original `Row` element:

- *For new rows:* no field attributes. If the function is called for a new row, it returns an empty list.
- *For modified rows:* deleted field attributes. If the function is called for a modified row, it returns those fields from the corresponding `OriginalRow` element that are not listed under the `Row` element.
- *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.

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 that this will only work if `OriginalRowSet` was enabled on the page source!

Usage
```
mt-db-deleted-original-rows($DB1)
```

▼ mt-db-modified-fields

Description
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.

Usage
```
mt-db-modified-fields($DB1/DB/RowSet/Row[3])
```

▼ mt-db-modified-rows

Description
Returns a list of the attributes of all the `Row` elements that were modified. The function can be used to determine the modifications to data read from the DB. Note that this will only work if `OriginalRowSet` was enabled on the page source!

Usage
```
mt-db-modified-rows($DB1)
```

▼ mt-db-new-fields

Description
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.

Usage
```
mt-db-new-fields($DB1/DB/RowSet/Row[1])
```

▼ mt-db-new-rows

Description
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 that this will only work if `OriginalRowSet` was enabled on the page source!

Usage
```
mt-db-new-rows($DB1)
```

▼ 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 and Share 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. 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-geolocation-started

Description

Returns `true()` if the solution has started geolocation tracking, `false()` otherwise.

Usage

```
mt-geolocation-started()
```

mt-has-serveraccess

Description

Returns `true` if server access is possible, `false` otherwise. 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.

Usage

```
mt-has-serveraccess(TimeoutSeconds as integer)
```

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`**

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, then the result text string is generated in the default **`'UTF-8'`** encoding. It is an error if no

`Encoding` argument s specified.

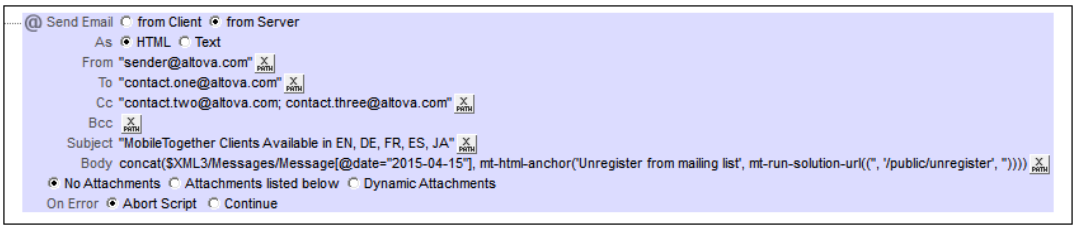
Examples

- **`mt-hexBinary-to-string`**`('48656C6C6F20576F726C64', 'ASCII')` returns `'Hello World'`
- **`mt-hexBinary-to-string`**`('48656C6C6F20576F726C64', '')` returns `'Hello World'`
- **`mt-hexBinary-to-string`**`('48656C6C6F20576F726C64')` returns an error

mt-html-anchor

Description

The **`mt-html-anchor`** 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 action. The link can open a an Internet page or a MobileTogether solution. To add a link to the email body, use the **`mt-html-anchor`** function in the XPath expression of the *Body* option (*see screenshot below*).



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, (ii) the height.

Examples

- **`mt-image-width-and-height`**`($XML1/images/png)` returns `364 76`

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-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, Load/Save File, Load/Save Binary File, Load Image. Note that operating systems other than Android might not support this function.

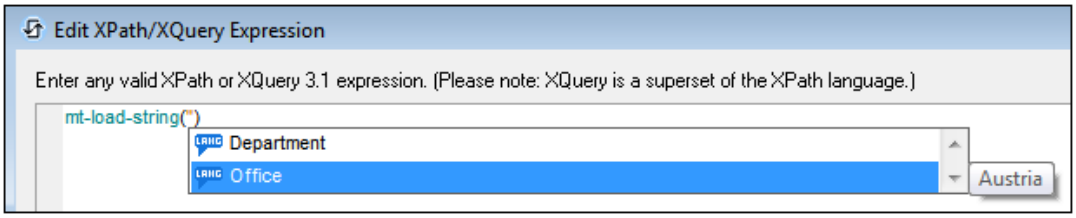
Usage
```
mt-last-file-path()
```

mt-load-string

Description
Returns the custom string identified by the `StringName` argument. Each custom string is part of a pool of strings that are defined in the Localization dialog. 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.

Usage
```
mt-load-string('StringName')
```

When the `mt-load-string` function is being entered in the Edit XPath/XQuery Expression dialog, 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**.



Cycle through the list by using the **Up** and **Down** keyboard buttons. The value of the selected custom string is displayed to the right of the popup (*see screenshot above*). The localization language of the displayed value is that of the simulation language that is currently selected in MobileTogether Designer. To enter the name of a custom string in the XPath expression, select the string, or cycle through the custom string 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 and **Project | Localization** 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, false() otherwise.

Usage
```
mt-nfc-started()
```

▼ 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.

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?**
Generates the URL of a MobileTogether AppStore App from the three submitted arguments. The URL will start the app from a hyperlink (that would typically be sent in an email). The hyperlink's target URL must have the format: **<url-scheme>://<url-host>**. The scheme information will be stored in the app's manifest file. It indicates to the device that this app should 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 open this AppStore App. See the section AppStore Apps for details.

- `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).
- `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).
- `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 . 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** 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 the URL of a solution from the three submitted arguments:

- `ServerAddress`: Takes the name or IP address of the MobileTogether Server on which the solution that you want to run is deployed
- `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)

- `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 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`** 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 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-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:

- If the first argument, `ServerAddress`, is the empty string, then the current server is used.
- The first `ServerAddress` argument is used to look up server information stored on the client. 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.
- If the second argument, `SolutionName`, is the empty string, then the current solution is used.
- 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 another <%MT%>-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?`**
The **`mt-run-solution-url-parameters`** function is intended for use as the third argument of the **`mt-run-solution-url`** function. The argument of the **`mt-run-solution-url-parameters`** function is a sequence of string values. These are the parameter values of the query string that will be generated by the **`mt-run-solution-url`** function. 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.

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>, '"', "'")`**.

The values of these parameters can be accessed in other design components by using the **`$MT_InputParameters`** global variable.

▣ *Example*

- **`mt-run-solution-url-parameters`**`(('2015', 'USA', 'true'))` returns `'&in1=2015&in2=USA&in3=true'`

▼ 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-string-to-hexBinary

mt-string-to-hexBinary(Text *as xs:string,* **Encoding** *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, then the default encoding **'UTF-8'** is used. It is an error if no **Encoding** argument s specified.

▣ *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 an error

▼ mt-test-case-run

mt-test-case-run() as **map(*)**
Returns a map containing information about the currently executed test run. 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-avaialable("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 is in progress, `false()` otherwise.

▼ mt-transform-image

See the description of this function in the Image-related Functions 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.

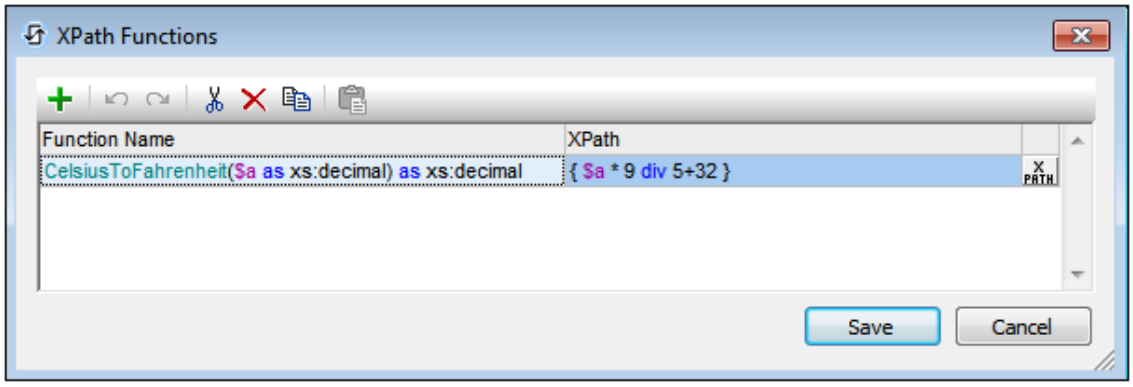
▼ mt-wait-cursor-shown

Description
Returns true()if the client device is showing its Wait cursor, false() otherwise.

Usage
mt-wait-cursor-shown()

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**. 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.

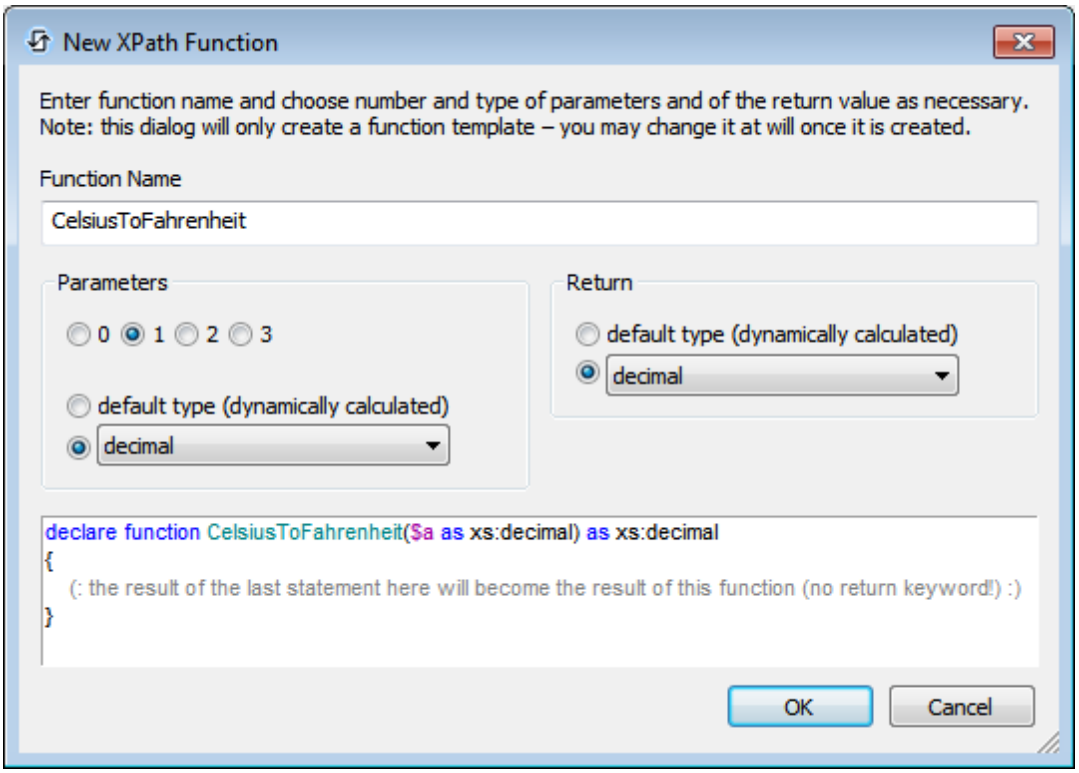


The function list can be ordered by function name. Do this by clicking the header of the Function Name column. Each click cycles the ordering through the following ordering sequence: (i) ascending, (ii) descending, (iii) dialog order. The order in the dialog can be modified by dragging-and-dropping functions 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.

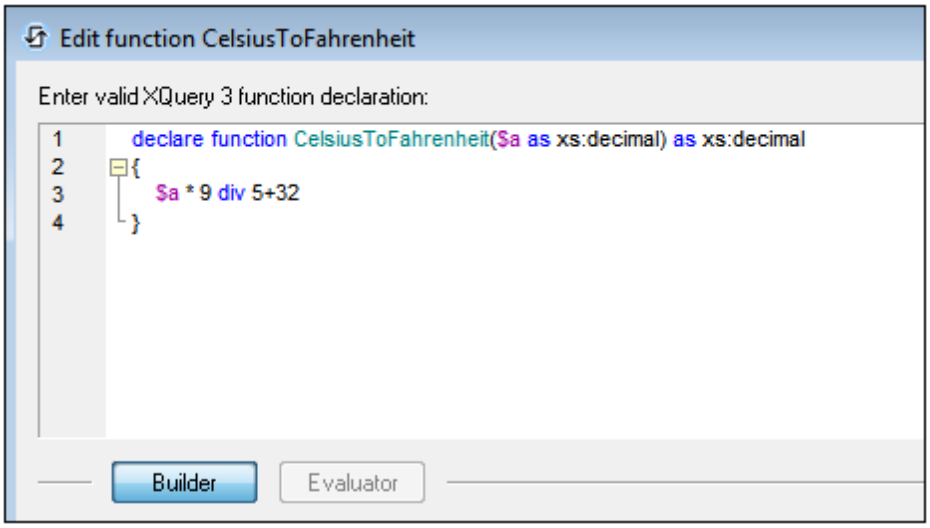
Add a new user-defined XPath function

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*).



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.



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 is used for all XPath/XQuery functions, including extension functions and user-defined functions. In order to avoid ambiguities involving built-in functions, we recommend that you capitalize user-defined functions.

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?

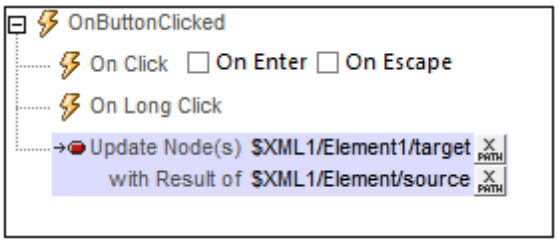
If your XPath expression is inside the repeating row of a table, then the element corresponding to the row of the table is the context node, say, `Row`.

- 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. Consider the Update Node(s) action defined in the screenshot below.



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>`.

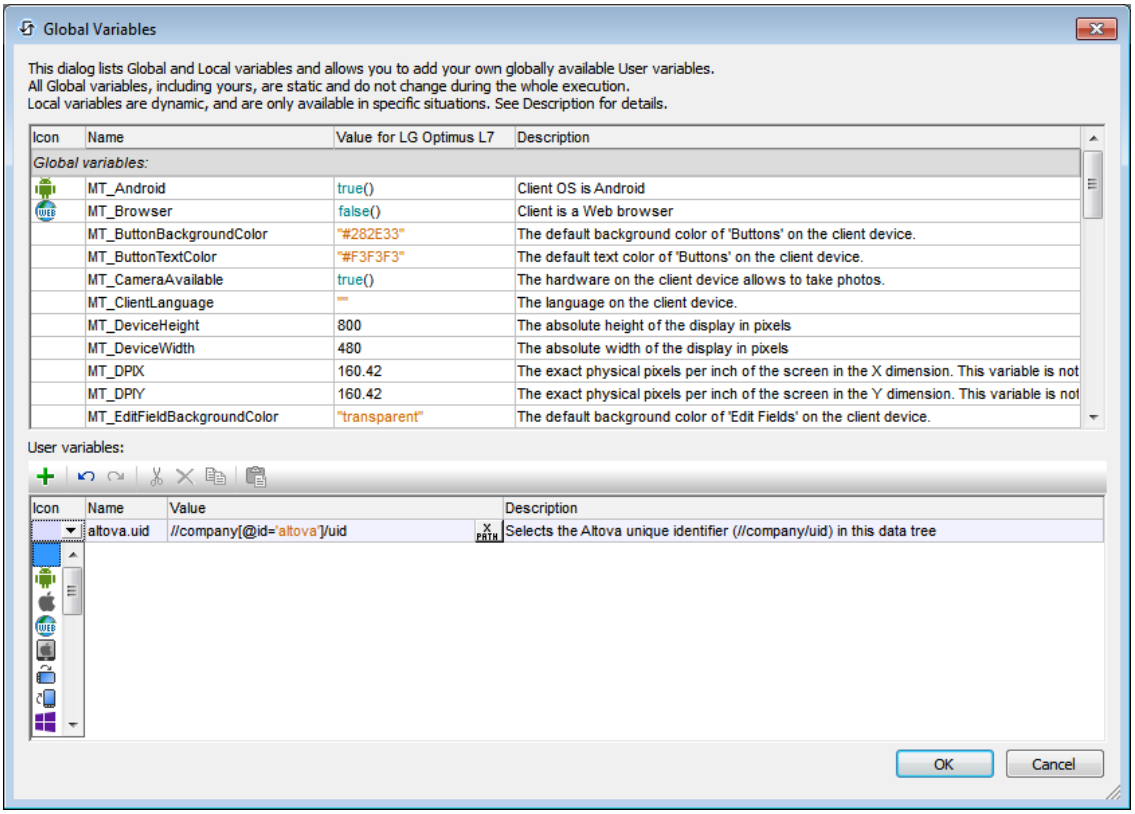
Note: If you wish to include the text content of descendant node/s, use a **`string`** function. Using the XML example above, for instance, the expression **`string($XML1/Element1/source, '')`** will return **`"AAABBB"`**.

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.

12.2 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.



The Global Variables dialog (*screenshot above*) displays three types of variables:

- Static-value variables: 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. The values of variables vary with the client device. For example, the variable **`$MT_Android`** has a value of **`true()`** when the mobile device being used is an Android.
- Dynamic-value variables: These variables contain device-related and project-related values that can change during execution. For example, the **`$MT_ControlNode`** variable has different values according to which node is the current node at a given time during project

execution.

- User variables: 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')
```

Static Global Variables

Static-value variables are called *Global Variables* in the Global Variables dialog. 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 (**Project | Global Variables**). In the dialog, the header of the *Value* column displays the mobile device that is selected in the Device Selector combo box. 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 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 SMS and telephony services are available on the mobile device, and can be used to make checks before initiating SMS or call actions. 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.

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 a camera application and geolocation tracking are available on the mobile device. They can be used to make checks before initiating image-taking, geolocation-related, or NFC-related 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_GeolocationAvailable`	`true() | false() | undefined`
`MT_NFCAvailable`	`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_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. Currently, parameter values are passed to the solution when a hyperlink to the solution is clicked. 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 **MT_InputParameters** variable holds parameter values as a sequence of string-value items. To retrieve a single parameter value, 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]**. For more information about hyperlinking and the `MT_InputParameters` variable, see Hyperlinking to Solutions.

- MT_IsAppStoreApp

  Indicates whether the current solution is running as an AppStore App 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 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 option 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 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.

Dynamic Local Variables

Dynamic-value variables are called *Local Variables* in the Global Variables dialog. 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 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. 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 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** and **MT_DeviceWidth**). See the note *Points Versus Pixels on iOS devices* below.

MT_CanvasX	*Length value in pixels*
MT_CanvasY	*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 **points** in the viewport coordinate space to **pixels** in the device coordinate space. In this way, design components maintain the same size relationship to the canvas and to each other, no matter what the resolution of the iOS device is.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions. For iOS devices, the values returned by the variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted to **points** in the viewport coordinate space. The respective numeric values are returned by the two variables as pixels for use in the design. You can calculate the relative sizes you require based on the values returned by the **$MT_CanvasX** and **$MT_CanvasY** variables. 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 **$MT_CanvasX * 0.5**. The XPath expression for the image width would be **concat($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	*Length value in pixels*
MT_WindowWidth	*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 data source node (the source node 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	*String value*
MT_ControlName	*String value*
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. 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. It holds an integer that is the number of the audio channel (`1 to 5`) that triggered the event.

  - MT_DBExecute_Result

    The XML result of the last executed DB Execute action. Note that any kind of SQL statement can be used in the DB Execute action. 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_FirstPageLoad

    Set to `true()` if this is the first time the page is loaded during the current workflow execution.

  - MT_HTTPExecute_Result

    The XML result of the last executed Execute SOAP Request or Execute REST Request.

  - MT_PageName

    The name of the page.

  - MT_ServerConnectionErrorLocation

    This variable is a sequence of strings that contains the action stack that triggered the `OnServerConnectionError` 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.

- MT_TargetNode

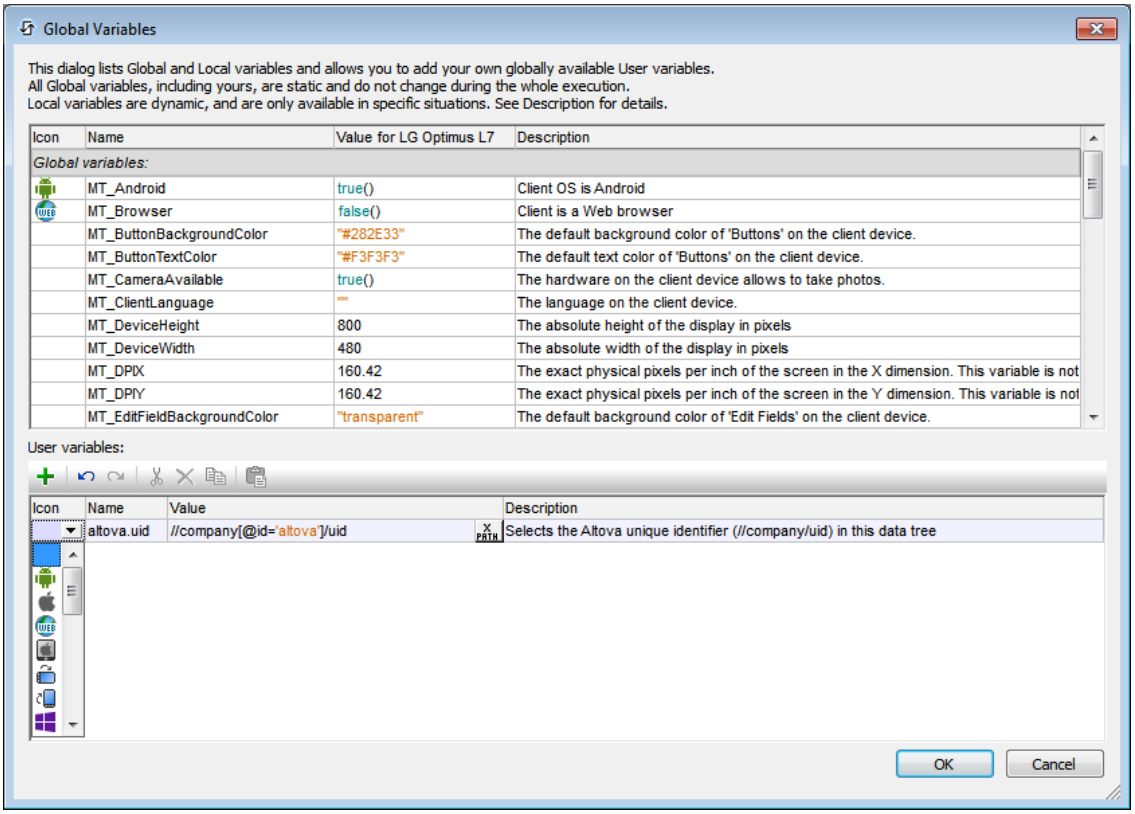
  This variable identifies the target node of an Update Node(s), Insert Node(s), Append Node(s), or Replace Node(s) action. It can be used to generate update values and new node properties according to what the target node is. See the descriptions of the respective actions for examples of how the variable can be used. `$MT_TargetNode` can also be used with the DB Execute action and the **Ensure Exists on Load (XPath)** command.

- 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.

User Variables

User Variables are variables that you define in the lower pane of the Global Variables dialog (**Project | Global Variables**, *screenshot below*).



To add a user variable, in the lower pane, do the following:

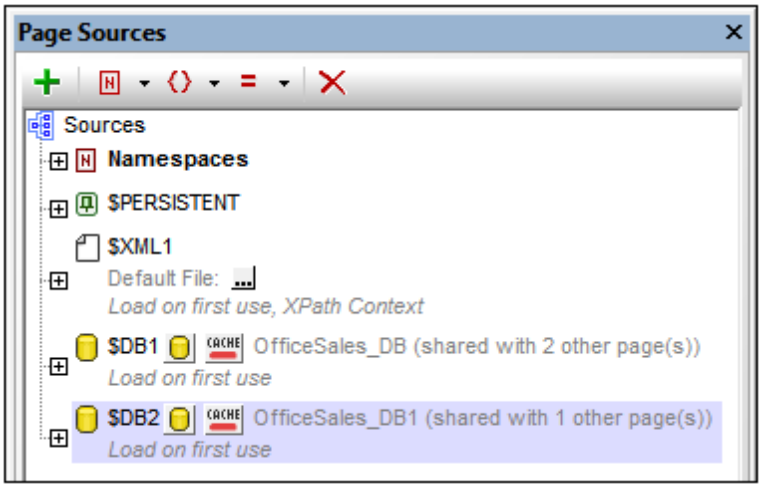
1. Click the **Append** or **Insert** icons (located in the pane's toolbar) to add a new entry to the list.
2. Enter the name of your new variable (in the *Name* column, and without a **$** symbol) and give the variable a description (*Description* column). *See screenshot above*.
3. Click in the *Value* field to bring up the Edit XPath/XQuery Expression dialog, and enter the XPath expression that determines the value of the 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.

Chapter 13

Databases

13 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, edited, and saved using a variety of mechanisms, including XQuery expressions.



This section

This section is organized as follows:

- DBs as Data Sources
- Connecting to a DB
- Selecting DB Objects as Data Sources
- Editing DB Data
- Saving Data to the DB
- The DB Execute Action
- Displaying DB Data
- Database Query

See the database tutorial, Database-And-Charts, 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 following databases are supported. The available root object for each database is also listed. While Altova endeavors to support other databases, successful connection and data processing have only been tested with the databases listed below. 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	Root Object	Notes
Firebird 2.5.4	database	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	schema	
IBM DB2 for i 6.1, 7.1	schema	Logical files are supported and shown as views.
IBM Informix 11.70	database	
MariaDB 10	database	
Microsoft Access 2003, 2007, 2010, 2013	database	
Microsoft Azure SQL Database	database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016	database	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7	database	
Oracle 9i, 10g, 11g, 12c	schema	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6	database	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	database	
SQLite 3.x	database	SQLite connections are supported as native, direct connections to the SQLite database file. No separate drivers are required.
Sybase ASE 15, 16	database	
Teradata 16	database	

For connecting to a SQLite DB, use MobileTogether Designer's Connection Wizard.

Note
On Linux and macOS servers, the only database connections supported are JDBC.

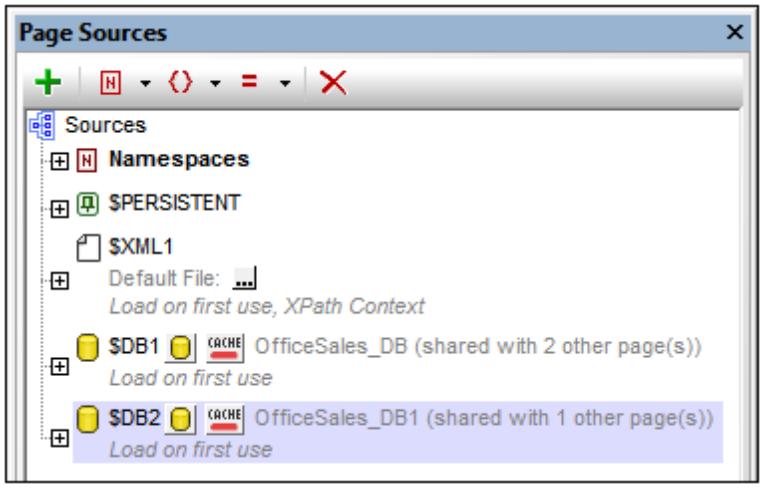
13.1 DBs as Data Sources

This section:

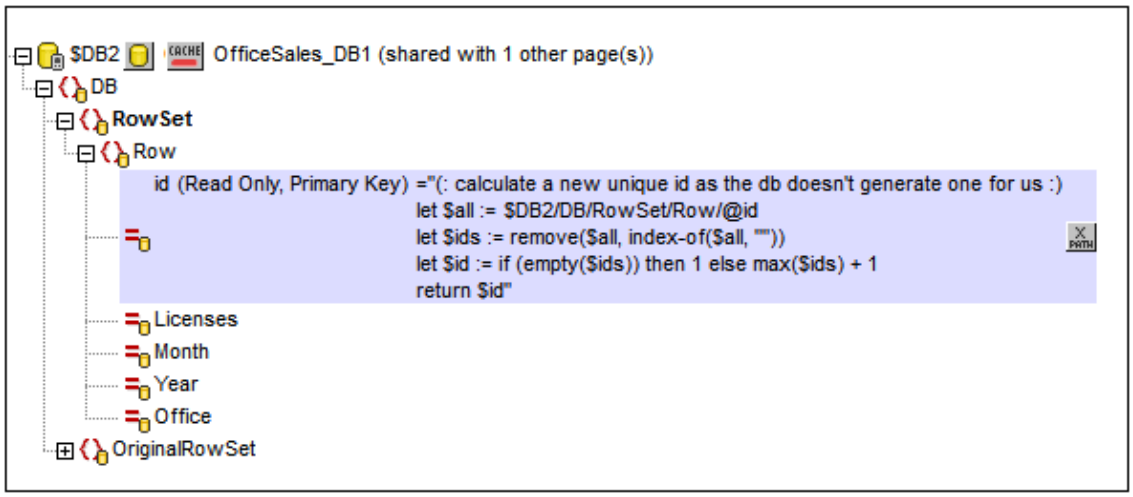
- About DB data sources
- Tree structure of the DB data source
- Switching data sources
- About OriginalRowSet
- Primary Keys in MobileTogether Designer

About DB data sources

Any number of DB data sources can be added to the design of a page and then used within it. Whether a DB data source is editable or not is defined at the time the data source is added. Specify a DB data source to be uneditable if its data is required only for presentation. Make the data source editable if you want to allow clients to modify DB data.



When a DB source is added, a data tree is generated (*see screenshot below and the section Tree structure below*). Each DB table row corresponds to a Row element; the table's columns are added as attributes of the Row element. If the data source is used on multiple pages, then a single tree structure can be shared across all instances of the data source. The option to share a tree structure is available each time a data source is added that is used on another page. When a shared structure is modified, you are offered the option of modifying the shared data source in its multiple instances across pages; alternatively, the shared data source is modified only in the instance in which it is modified..



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 data sources

After you have created a design that uses a DB as a data source, you can switch the data source to another DB that has the same structure and continue to use the original design. To switch the DBs of a data source, right-click the `$DB` root node of the tree, select **Choose DB Data Source**, and continue with the DB connection process.

Two DBs are considered to have the same structure if the 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 data source will not be updated with data from the new DB. If the DB switch is aborted, then the data 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 data source

Every DB data 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)
```

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 ca 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

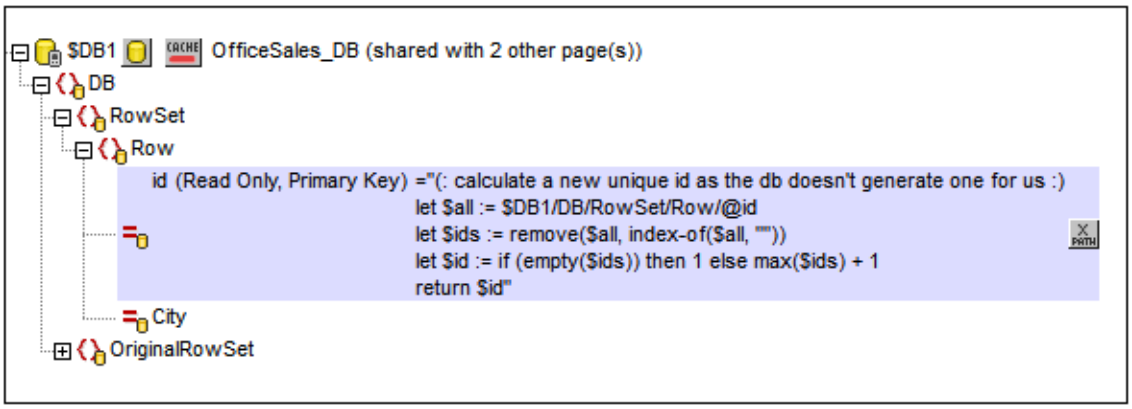
In order for data to be edited and saved, the tree of the page source must also include an `OriginalRowSet` element, which is a copy of the `RowSet` element. The original data is saved in the `OriginalRowSet` element, while edited data is saved in the `RowSet` element. When the page source is saved, the difference between the two trees, `OriginalRowSet` and `RowSet`, is calculated, and the data 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.

To create an `OriginalRowSet` for a page source, right-click the root node of the page source and toggle on the command **Create OriginalRowSet**.

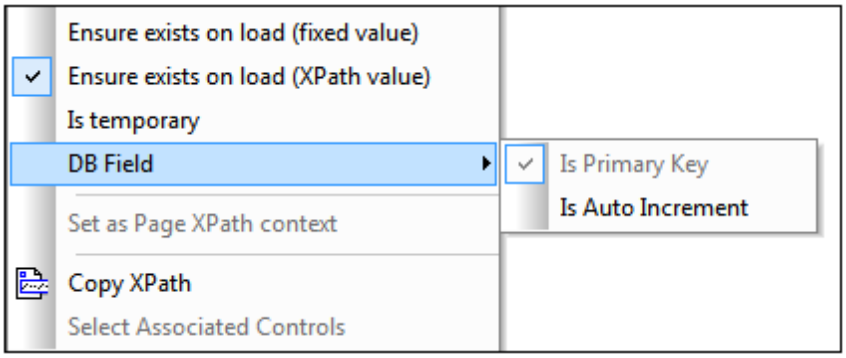
The **Create OriginalRowSet** command is enabled for database type (`$DB`) root nodes. It is a toggle command that creates/removes an `OriginalRowSet` data structure that contains the original data of the page source. User modified data is saved in the main structure created from the data source. When modified data is saved back to the DB, the `OriginalRowSet` structure is modified to contain the data newly saved to the DB. 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.

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*).

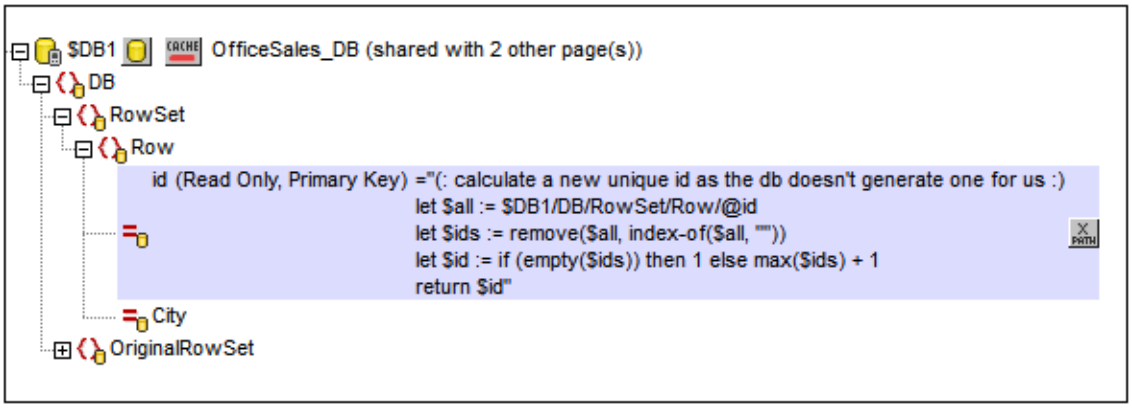


If auto-retrieval of this information was not successful, the context menu of tree nodes contains toggle commands that enable you to correctly annotate nodes (*see screenshot below*).



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
```

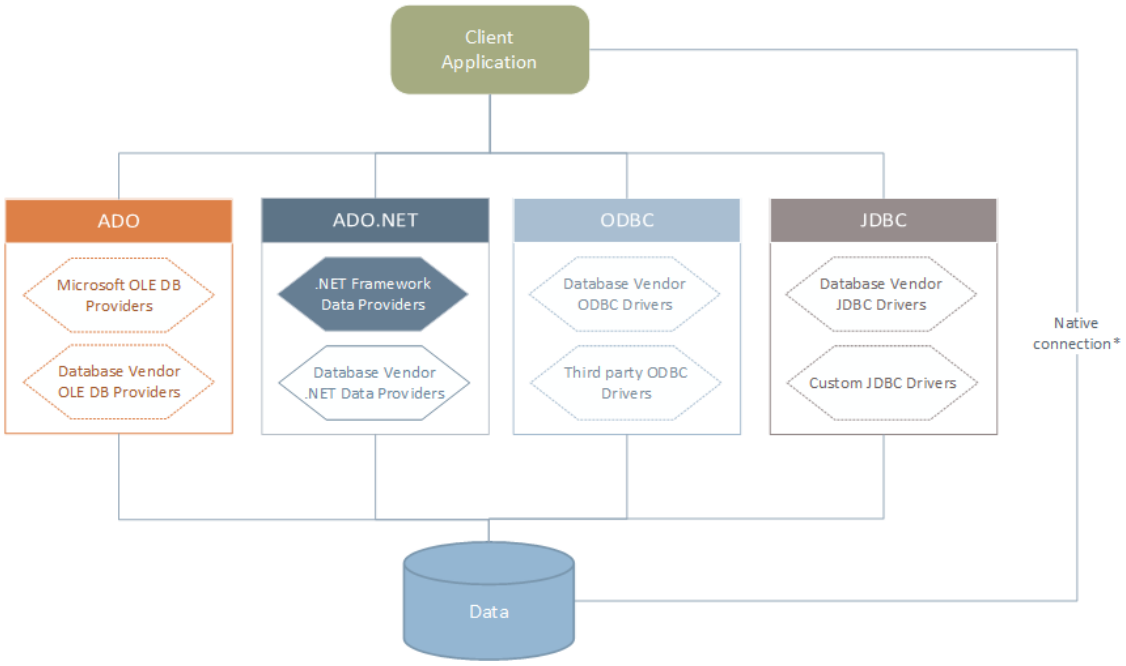


13.2 Connecting to a Database

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 following diagram illustrates, in a simplified way, 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).



** Direct native connections are supported for SQLite and PostgreSQL databases. To connect to such databases, no additional drivers are required to be installed 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)

Some ADO.NET providers are not supported or have limited support. See ADO.NET Support Notes.

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.

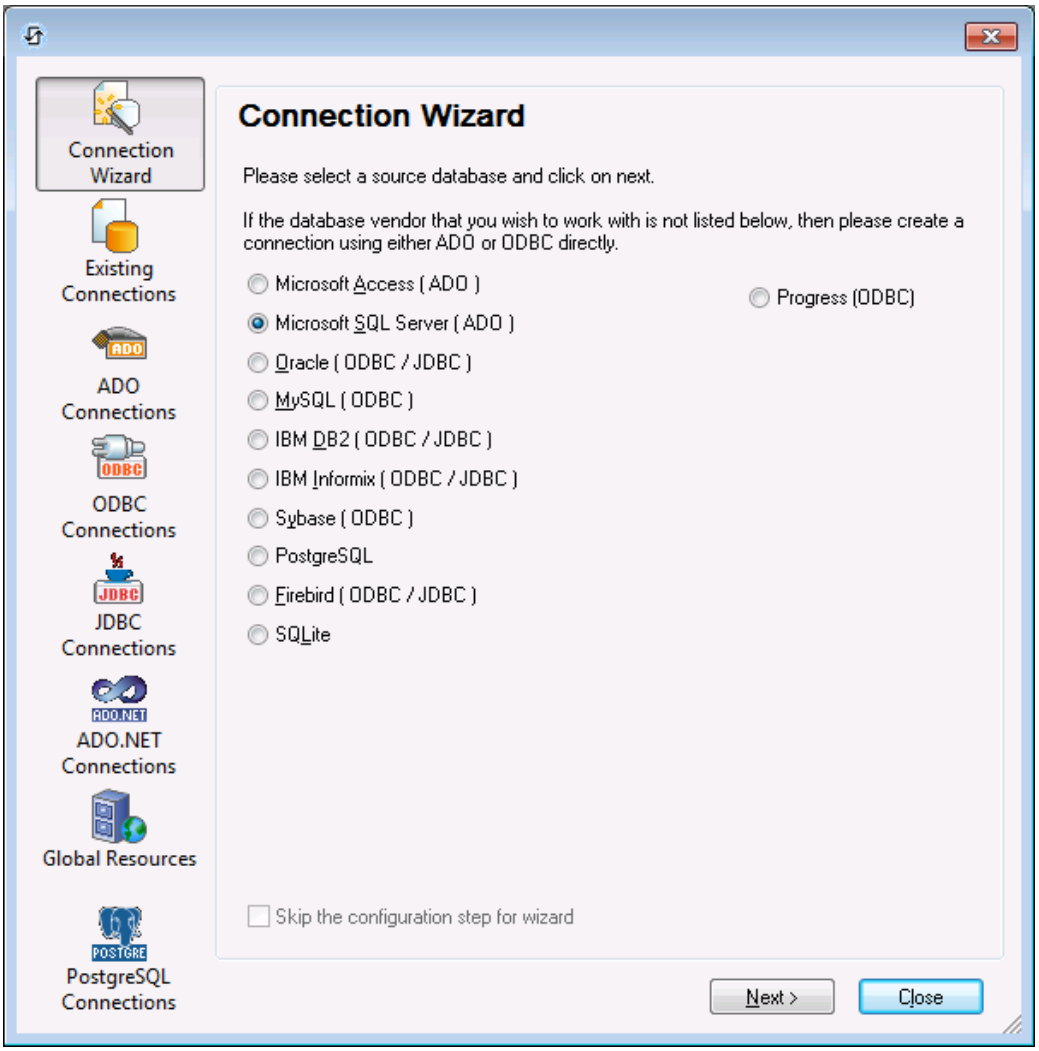
Starting the Database Connection Wizard

Whenever you take an action that requires a database connection, a wizard appears that guides you through the steps required to set up the connection.

Before you go through the wizard steps, be aware that for some database types it is necessary to install and configure separately 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.

To start the database connection wizard:

- 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.



After you select a database type and click **Next**, the on-screen instructions will depend on the

database kind, technology (ADO, ADO.NET, ODBC, JDBC) and driver used.

For examples applicable to each database type, see Database Connection Examples. For instructions applicable to each database access technology, refer to the following topics:

- Setting up an ADO Connection
- Setting up an ADO.NET Connection
- Setting up an ODBC Connection
- Setting up a JDBC Connection

Database Drivers Overview

The following table lists common database drivers you can use to connect to a particular database through a particular data access technology. Note that this list does not aim to be either exhaustive or prescriptive; you can use other native or third party alternatives in addition to the drivers shown below.

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. For some databases, the latest driver supplied by the database vendor is likely to perform better than the driver that shipped with the operating system.

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, it is strongly recommended to read carefully 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.
- 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.
- When setting up an ODBC data source, it is recommended to create the data source name (DSN) as System DSN instead of User DSN. For more information, see Setting up an ODBC Connection.
- 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.
- 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.

Database	Interface	Drivers
Firebird	ADO.NET	Firebird ADO.NET Data Provider (http://www.firebirdsql.org/en/additional-downloads/)
	JDBC	Firebird JDBC driver (http://www.firebirdsql.org/en/jdbc-driver/)

Database	Interface	Drivers
	ODBC	Firebird ODBC driver (http://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 • IBM DB2 for i5/OS IBMDARLA OLE DB Provider • 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 • 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 (http://dev.mysql.com/downloads/connector/net/).
	JDBC	MariaDB Connector/J (https://downloads.mariadb.org/)
	ODBC	MariaDB Connector/ODBC (https://downloads.mariadb.org/)
Microsoft SQL Server	ADO	• Microsoft OLE DB Provider for SQL Server • SQL Server Native Client
	ADO.NET	• .NET Framework Data Provider for SQL Server • .NET Framework Data Provider for OLE DB
	JDBC	• Microsoft JDBC Driver for SQL Server (http://msdn.microsoft.com/en-us/data/aa937724.aspx)
	ODBC	• SQL Server Native Client
MySQL	ADO.NET	Connector/NET (http://dev.mysql.com/downloads/connector/net/)
	JDBC	Connector/J (http://dev.mysql.com/downloads/connector/j/)

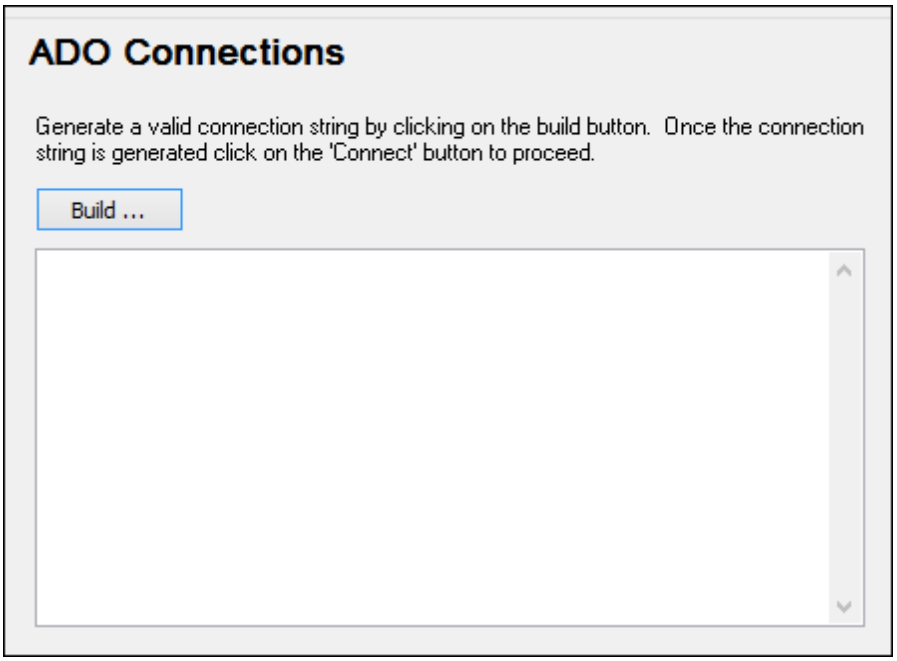
Database	Interface	Drivers
	ODBC	Connector/ODBC (http://dev.mysql.com/downloads/connector/odbc/)
Oracle	ADO	• Oracle Provider for OLE DB • Microsoft OLE DB Provider for Oracle
	ADO.NET	Oracle Data Provider for .NET (http://www.oracle.com/technetwork/topics/dotnet/index-085163.html)
	JDBC	• JDBC Thin Driver • JDBC Oracle Call Interface (OCI) Driver 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 • 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. There is no need to install any drivers if using native connection.
Progress OpenEdge	JDBC	JDBC Connector (https://www.progress.com/jdbc/openedge)
	ODBC	ODBC Connector (https://www.progress.com/odbc/openedge)
SQLite	Native Connection	Available. There is no need to install any drivers if using native connection.
Sybase	ADO	Sybase ASE OLE DB Provider
	JDBC	jConnect™ for JDBC
	ODBC	Sybase ASE ODBC Driver
Teradata	ADO.NET	.NET Data Provider for Teradata (http://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata)
	JDBC	Teradata JDBC Driver (http://downloads.teradata.com/download/connectivity/jdbc-driver)
	ODBC	Teradata ODBC Driver for Windows (http://downloads.teradata.com/download/connectivity/odbc-driver/windows)

Setting up an ADO Connection

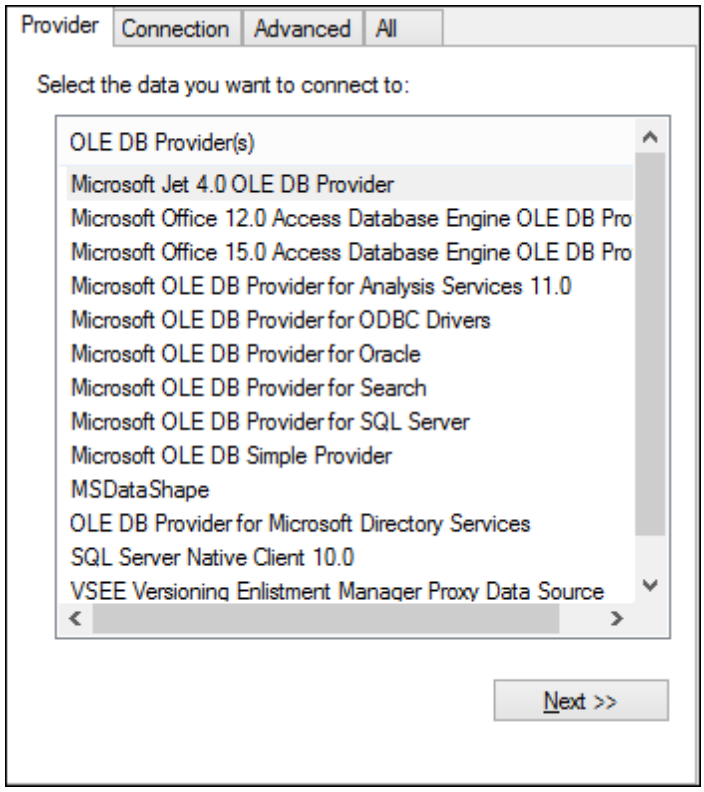
Microsoft ActiveX Data Objects (ADO) is a data access technology that enables you to connect to a variety of data sources through OLE DB. OLE DB is an alternative interface to ODBC or JDBC; it provides uniform access to data in a COM (Component Object Model) environment. ADO is the typical choice for connecting to Microsoft native databases such as Microsoft Access or SQL Server, although you can also use it for other data sources.

To set up an ADO connection:

1. Start the database connection wizard.
2. Click **ADO Connections**.



3. Click **Build**.



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** When connecting to Microsoft Access 2003, you can also use the **Microsoft Jet OLE DB Provider**.
SQL Server	• **SQL Server Native Client** • **Microsoft OLE DB Provider for SQL Server**
Other database	Select the provider applicable to your database. If an OLE DB provider to your database is not available, install the required driver from the database vendor (see Database Drivers Overview). Alternatively, set up an ODBC or JDBC connection. If the operating system has an ODBC driver to the required database, you can also use the **Microsoft OLE DB Provider for ODBC Drivers**.

5. 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, as well as the database username and password. For Microsoft Access, you will be asked to browse for or provide the path to the database file.

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. The following sections provide guidance on configuring the basic initialization properties for Microsoft Access and SQL Server databases:

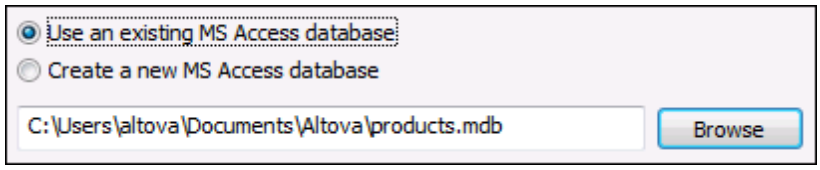
- Setting up the SQL Server Data Link Properties
- Setting up the Microsoft Access Data Link Properties

Connecting to an Existing Microsoft Access Database

This approach is suitable when you want to connect to a Microsoft Access database which is not password-protected. If the database is password-protected, set up the database password as shown in Connecting to Microsoft Access (ADO).

To connect to an existing Microsoft Access database:

1. Run the database connection wizard (see Starting the Database Connection Wizard).
2. Select **Microsoft Access (ADO)**, and then click **Next**.



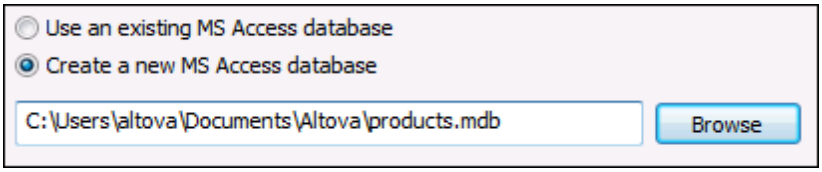
3. Select **Use an existing MS Access database**.
4. Browse for the database file, or enter the path to it (either relative or absolute).
5. Click **Connect**.

Creating a New Microsoft 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, even if Microsoft Access is not installed on the computer. The database file created by MobileTogether Designer is empty. To create the required database structure, use Microsoft Access or a tool such as DatabaseSpy (http://www.altova.com/databasespy.html).

To create a new Microsoft Access database:

1. Run the database connection wizard (see Starting the Database Connection Wizard).
2. Select **Microsoft Access (ADO)**, and then click **Next**.



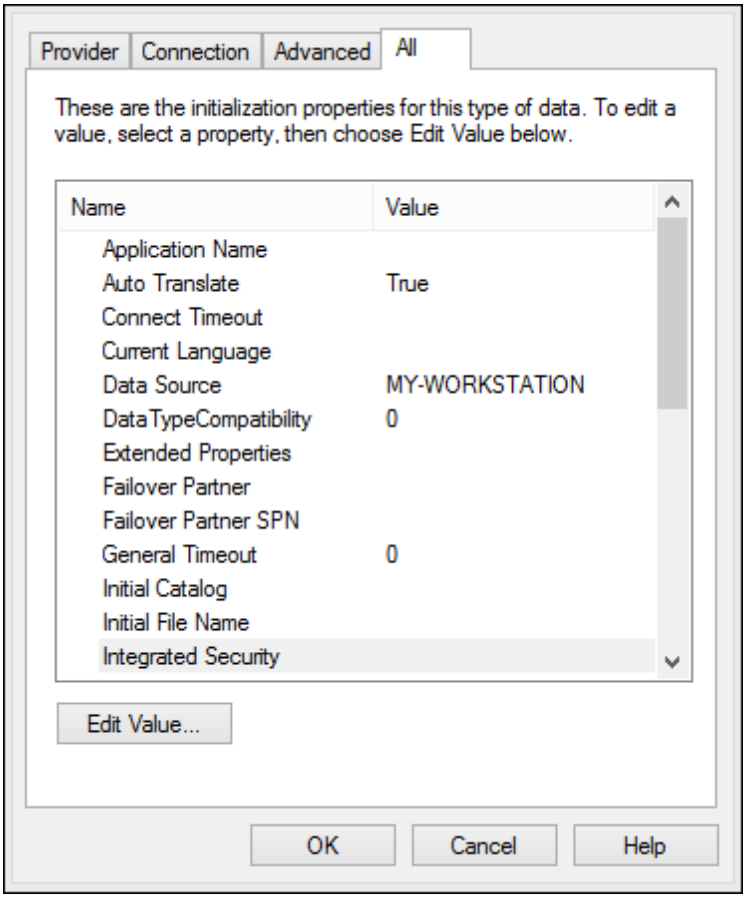
3. Select **Create a new MS Access database**, and then enter the path (either relative or absolute) of the database file to be created (for example, **c:\users\public\products.mdb**). Alternatively, click **Browse** to select a folder, type the name of the database file in the "File name" text box (for example, **products.mdb**), and click **Save**.

> **Notes**
> - Make sure that you have write permissions to the folder where you want to create the database file.
> - The database file name must have the **.mdb** or **.accdb** extension.

4. Click **Connect**.

Setting up the SQL Server Data Link Properties

When you connect to a Microsoft SQL Server database through ADO (see Setting up an ADO Connection), ensure that the following data link properties are configured correctly in the **All** tab of the Data Link Properties dialog box.

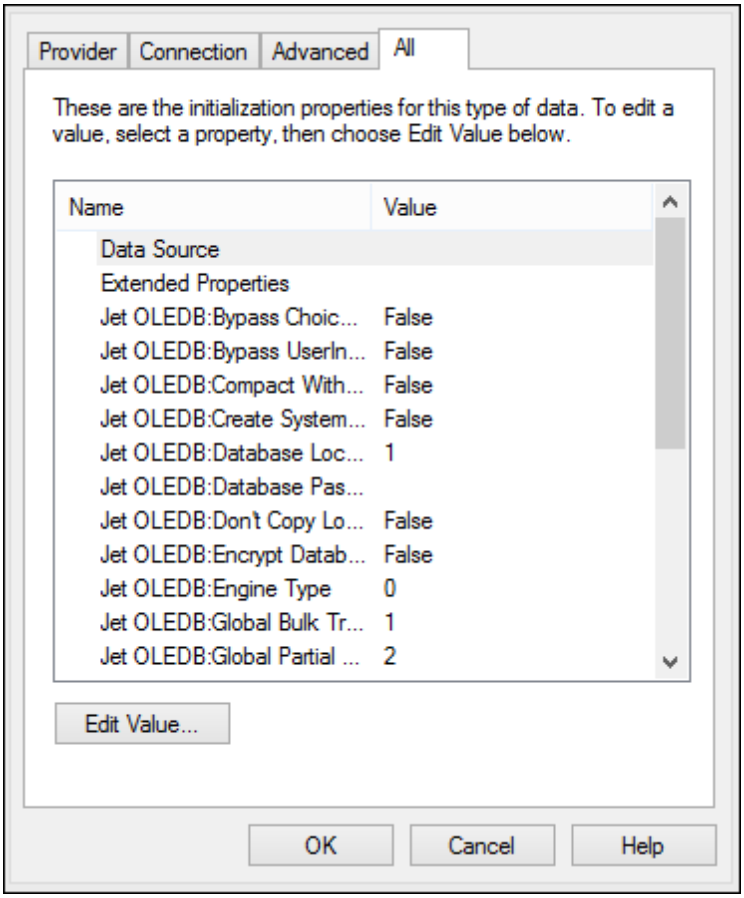


Data Link Properties dialog box

Property	Notes
Integrated Security	If you selected the **SQL Server Native Client** data provider on the **Provider** tab, set this property to a space character.
Persist Security Info	Set this property to **True**.

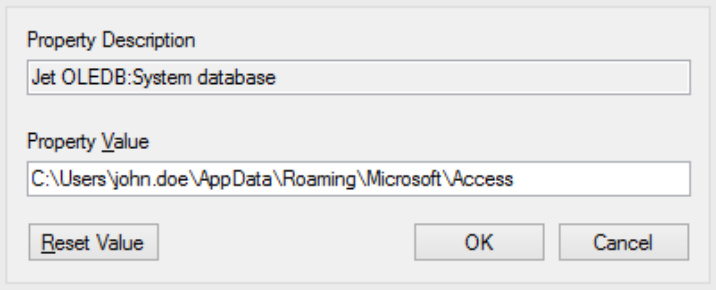
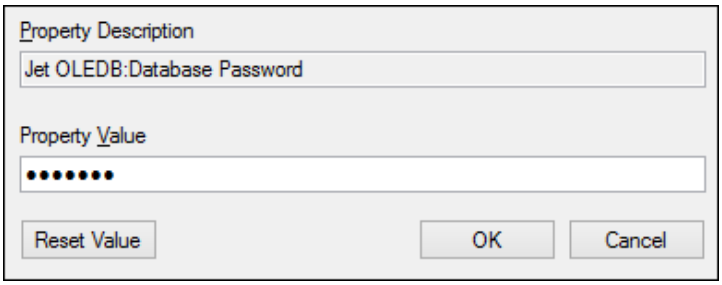
Setting up the Microsoft Access Data Link Properties

When you connect to a Microsoft Access database through ADO (see Setting up an ADO Connection), ensure that the following properties are configured correctly in the **All** tab of the Data Link Properties dialog box.



Data Link Properties dialog box

Property	Notes
Data Source	This property stores the path to the Microsoft Access database file. To avoid database connectivity issues, it is recommended to use 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

Property	Notes
	the path of the **System.MDW** file. 
Jet OLEDB:Database Password	If the database is password-protected, set the value of this property to the database password. 

Setting up an 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. As shown below, you 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. In other words, with ADO.NET, an application such as MobileTogether Designer interacts with a database through a data provider. Each data provider is optimized to work with the specific type of data source that it is designed for. There are two types of .NET providers:

1. Supplied by default with Microsoft .NET Framework.
2. 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.

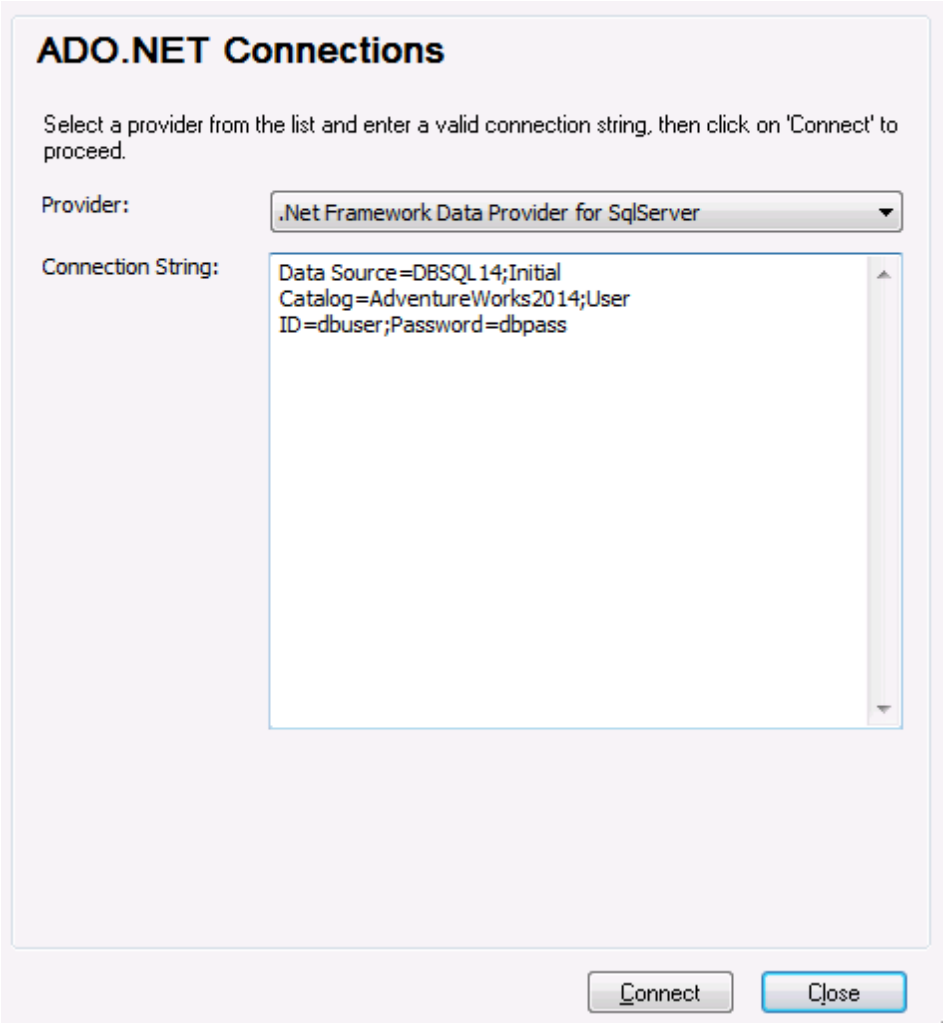
To set up an ADO.NET connection:

1. Start the database connection wizard.
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).

   The syntax of the connection string depends on the provider selected from the "Provider" list. For examples, see Sample ADO.NET Connection Strings.



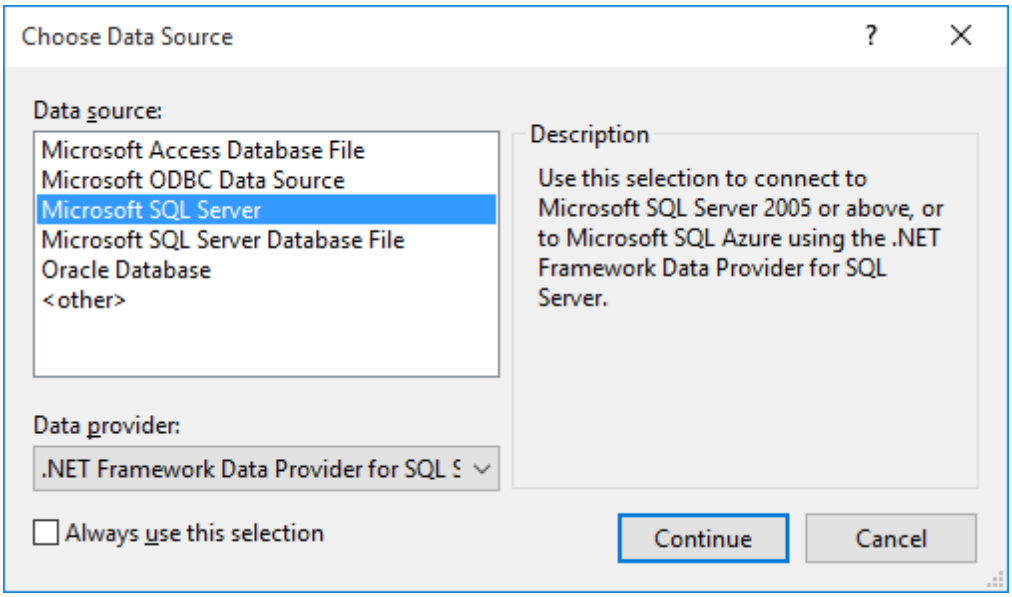
5. Click **Connect**.

Creating a Connection String in Visual Studio

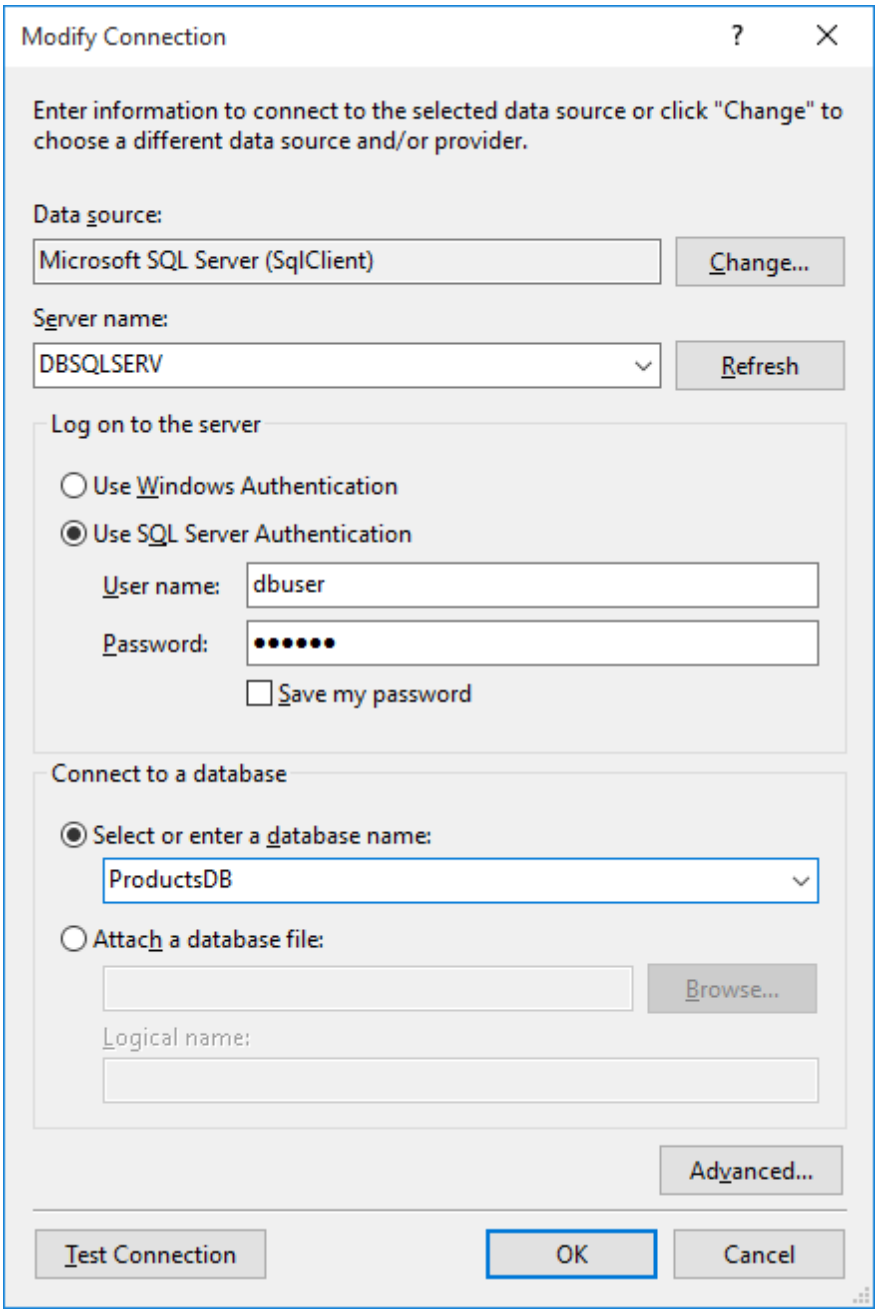
In order to connect to a data source using ADO.NET, a valid database connection string is required. The following instructions show you how to create a connection string from Visual Studio.

To create a connection string in 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.



3. Click **Continue**.



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.

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). 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 (http://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

```
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: http://www-01.ibm.com/support/docview.wss?uid=swg21429586.

See also: http://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 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).

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

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. See Setting up an ODBC Connection.
	.Net Framework Data Provider for OleDb	Limited support. Known issues exist with Microsoft Access connections. It is recommended to use ADO direct connections instead. See Setting up an ADO Connection.
Firebird	**Firebird ADO.NET Data Provider**	Limited support. It is recommended to use ODBC or JDBC instead. See Connecting to Firebird (ODBC) and Connecting to Firebird (JDBC).
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.
Sybase	**-**	No ADO.NET drivers for this vendor are supported.

Setting up an ODBC Connection

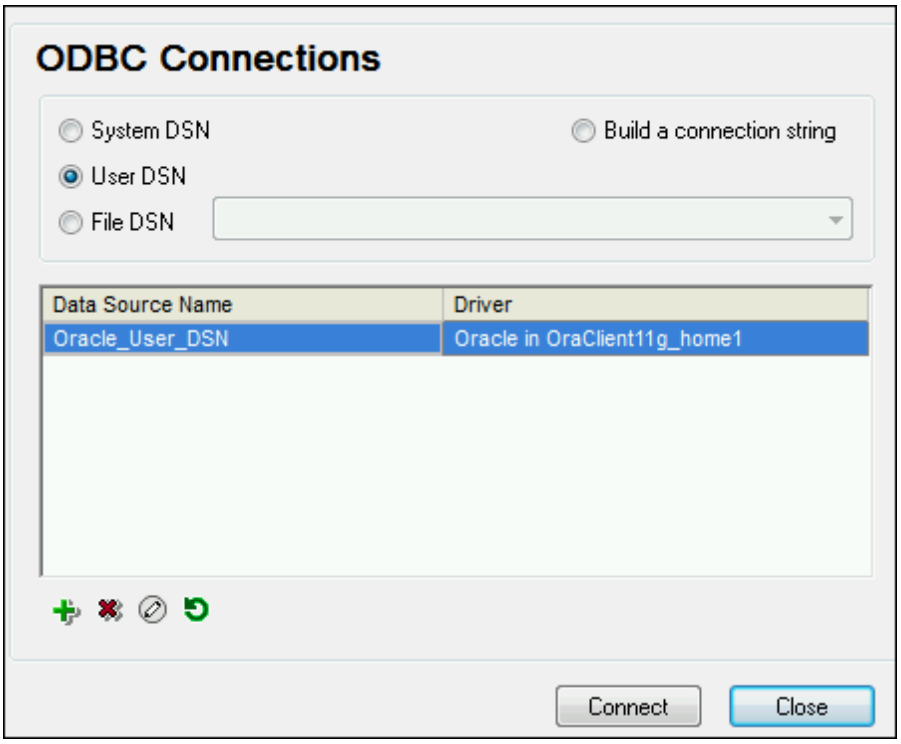
ODBC (Open Database Connectivity) is a widely used data access technology that enables you to connect to a database from MobileTogether Designer. It can be used either as primary means to connect to a database, or as an alternative to OLE DB- or JDBC-driven connections.

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.



ODBC Connections dialog box

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).

To connect by using a new DSN:

1. Start the database connection wizard.
2. On the database connection dialog box, click **ODBC Connections**.
3. Select a data source type (User DSN, System DSN, File DSN).

   > To create a System DSN, you need administrative rights on the operating system, and MobileTogether Designer must be run as administrator.

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).
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.

To connect by using an existing DSN:

1. Start the database connection wizard.
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**.

To build a connection string based on an existing .dsn file:

1. Start the database connection wizard.
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**.

To connect by using a prepared connection string:

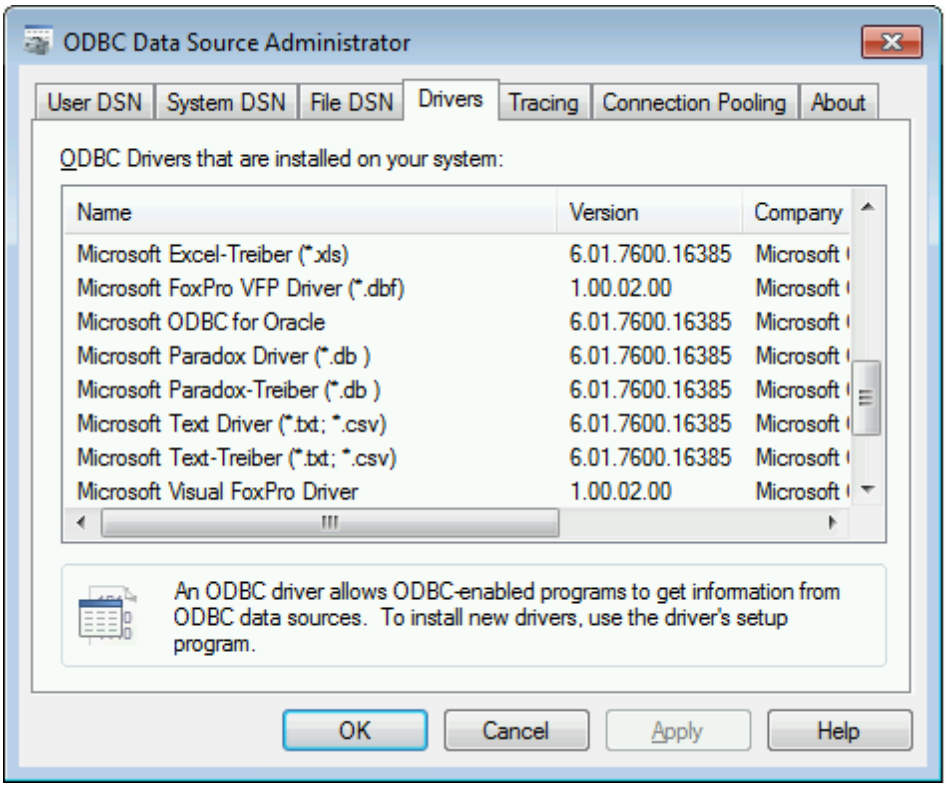
1. Start the database connection wizard.
2. Click **ODBC Connections**.
3. Select **Build a connection string**.
4. Paste the connection string into the provided box, and then click **Connect**.

Viewing the 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, 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 Data Source Administrator

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). Once the ODBC driver is available on your system, you are ready to create ODBC connections with it (see Setting up an ODBC Connection).

Setting up a JDBC Connection

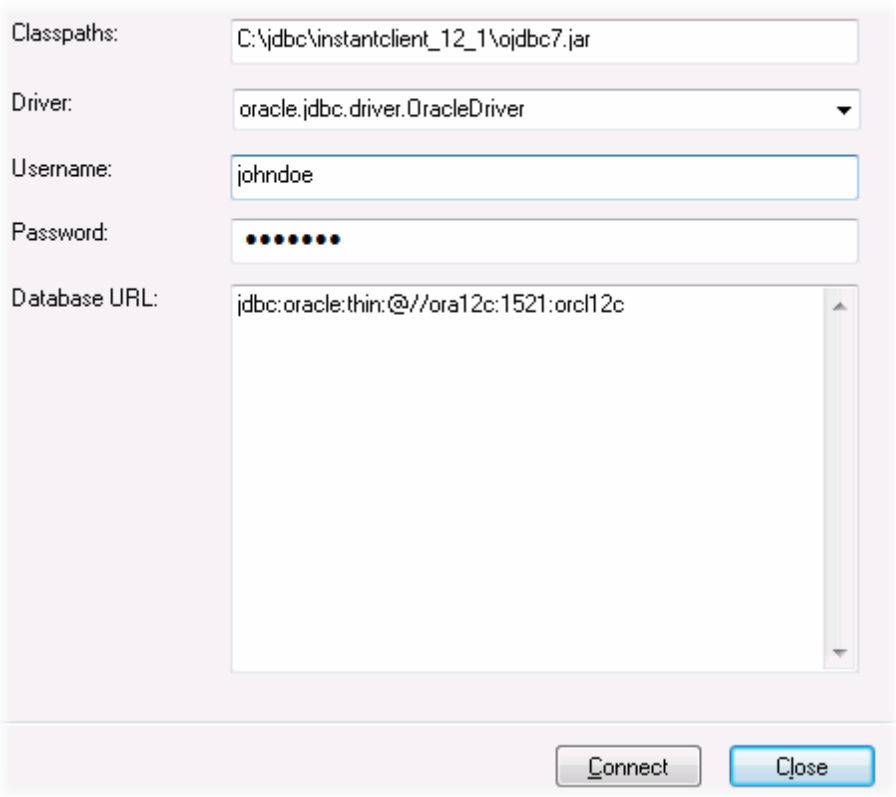
JDBC (Java Database Connectivity) is a database access interface which is part of the Java software platform from Oracle. JDBC connections are generally more resource-intensive than ODBC connections but may provide features not available through ODBC.

Prerequisites

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. If you have not installed it already, check the official Java website for the download package and installation instructions.
- The JDBC drivers from the database vendor must be installed. If you are connecting to an Oracle database, note that some Oracle drivers are specific to certain JRE versions and may require additional components and configuration. The documentation of your Oracle product (for example, the "Oracle Database JDBC Developer's Guide and Reference") includes detailed instructions about the configuration procedure for each JDBC driver.
- 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 `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. The documentation of the JDBC driver will typically include step-by-step instructions on setting the CLASSPATH variable (see also Configuring the CLASSPATH).

Setting up a JDBC connection

1. Start the database connection wizard.
2. Click **JDBC Connections**.
3. Optionally, enter a semicolon-separated list of .jar file paths in the "Classpaths" text box. The .jar libraries entered here will be loaded into the environment in addition to those already defined in the `CLASSPATH` environment variable. When you finish editing the "Classpaths" text box, any JDBC drivers found in the source .jar libraries are automatically added to the "Driver" list (see the next step).



4. Next to "Driver", select a JDBC driver from the list, or enter a Java class name. Note that this list contains any JDBC drivers configured through the CLASSPATH environment variable (see Configuring the CLASSPATH), as well as those found in the "Classpaths" text box.

   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 to 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.

5. Enter the username and password to the database in the corresponding boxes.
6. In the Database URL text box, enter the JDBC connection URL (string) in the format specific to your database type. The following table describes the syntax of JDBC connection URLs (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 Informix	`jdbc:informix-sqli://hostName:port/`

Database	JDBC Connection URL
	databaseName:INFORMIXSERVER=**myserver**
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**:**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: Syntax variations to the formats listed above are also possible (for example, the database URL may exclude the port or may include the username and password to the database). Check the documentation of the database vendor for further details.

7. Click **Connect**.

Configuring the CLASSPATH

The CLASSPATH environment variable is used by the Java Runtime Environment (JRE) 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 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 Informix	`C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;`
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;` 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`

- 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.

To configure the CLASSPATH on Windows 7:

1. Open the **Start** menu and right-click **Computer**.
2. Click **Properties**.
3. Click **Advanced system settings**.
4. In the **Advanced** tab, click **Environment Variables**,
5. Locate the CLASSPATH variable under user or system environment variables, and then click Edit. If the CLASSPATH variable does not exist, click **New** to create it.
6. Edit the variable value to include the path on your operating system where the JDBC driver is located. To separate the JDBC driver path from other paths that may already be in the CLASSPATH variable, use the semi-colon separator (;).

To configure the CLASSPATH on Windows 10:

1. Press the Windows key and start typing "environment variables".
2. Click the suggestion **Edit the system environment variables**.
3. Click **Environment Variables**.
4. Locate the CLASSPATH variable under user or system environment variables, and then click **Edit**. If the CLASSPATH variable does not exist, click **New** to create it.
5. Edit the variable value to include the path on your operating system where the JDBC driver is located. To separate the JDBC driver path from other paths that may already be in the CLASSPATH variable, use the semi-colon separator (;).

Setting up a PostgreSQL Connection

Connections to PostgreSQL databases can be set up either as native connections, or connections via ODBC, JDBC, and other drivers. The advantage of setting up a native connection is that no drivers are required to be installed on your system. Also, if you intend to deploy files for execution on a Linux or OS X server, no drivers are required to be installed on the target server as well (see also Database Connections on Linux and Mac).

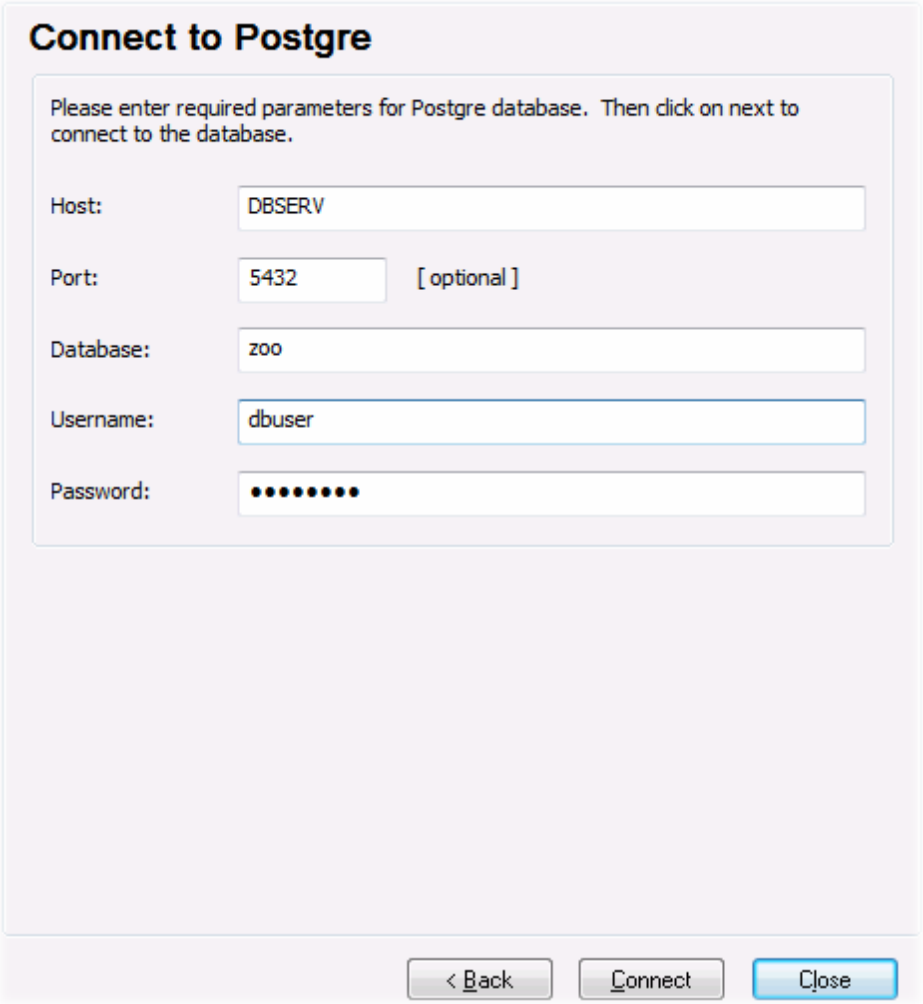
If you prefer to establish a connection by means of a non-native driver, see the following topics:

- Setting up a JDBC Connection
- Connecting to PostgreSQL (ODBC)

Otherwise, if you want to set up a native connection to PostgreSQL, follow the steps below. To proceed, you need the following prerequisites: host name, port, database name, username, and password.

To set up a native PostgreSQL connection:

1. Start the database connection wizard.
2. Click **PostgreSQL Connections**.
3. Enter the host (*localhost*, if PostgreSQL runs on the same machine), port (typically 5432, this is optional), the database name, username, and password in the corresponding boxes.



4. Click **Connect**.

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**.

Setting up a SQLite Connection

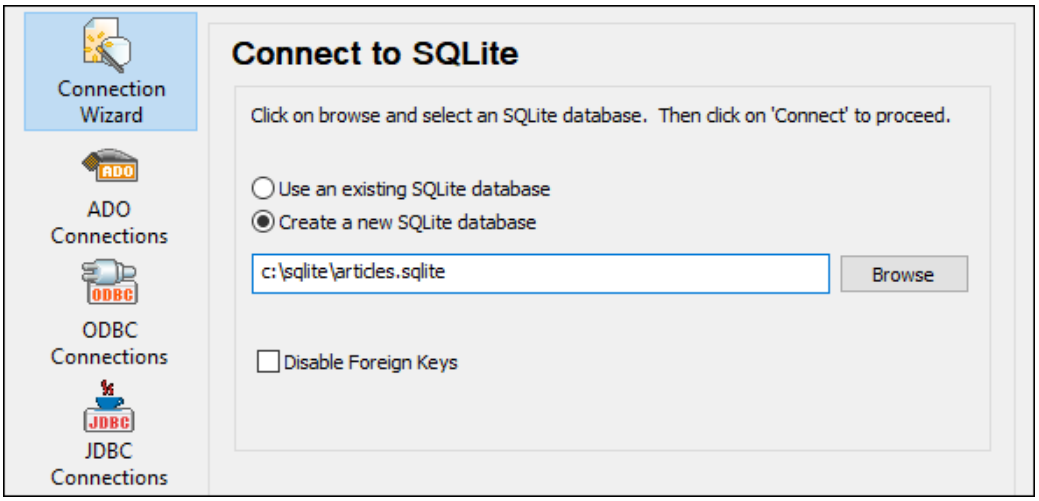
SQLite (http://www.sqlite.org) is a file-based, self-contained database type, which makes it ideal in scenarios where portability and ease of configuration is important. Since SQLite databases are natively supported by MobileTogether Designer, you do not need to install any drivers to connect to them.

Creating a New SQLite Database

You can create a new SQLite database file and connect to it, as an alternative to connecting to an existing database file. The database file created by MobileTogether Designer is empty; use queries or scripts to create the required database structure and populate it with data.

To create a new SQLite database:

1. Run the database connection wizard (see Starting the Database Connection Wizard).
2. Select **SQLite**, and then click **Next**.



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 (for example, **products.sqlite**), and click **Save**.

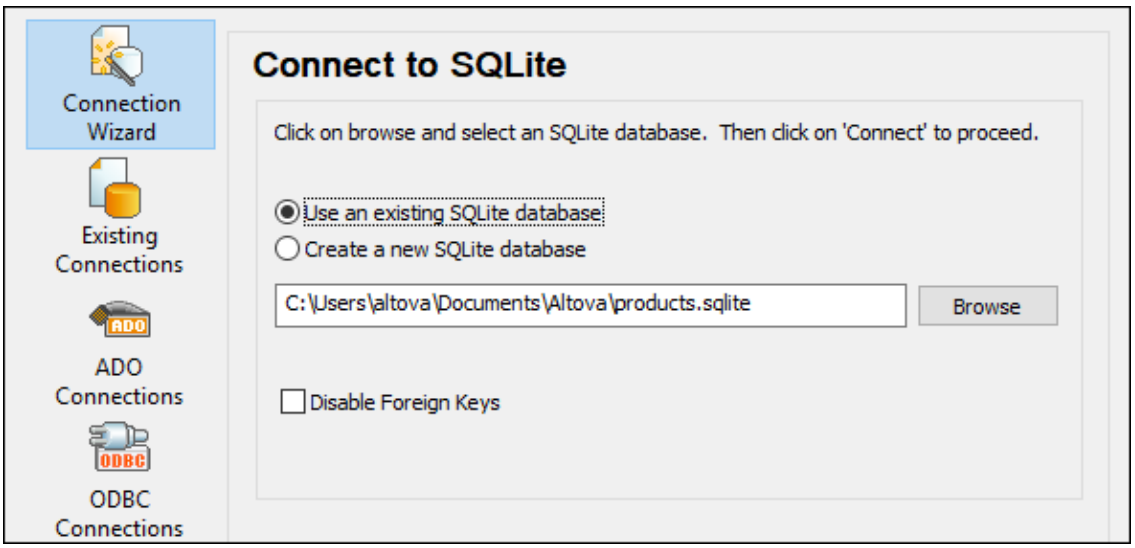
> Make sure that you have write permissions to the folder where you want to create the database file.

4. Optionally, select the **Disable Foreign Keys** check box, see Foreign Key Constraints.
5. Click **Connect**.

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" wizard page

When foreign keys are disabled, you will be able to perform operations against 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 in 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.)

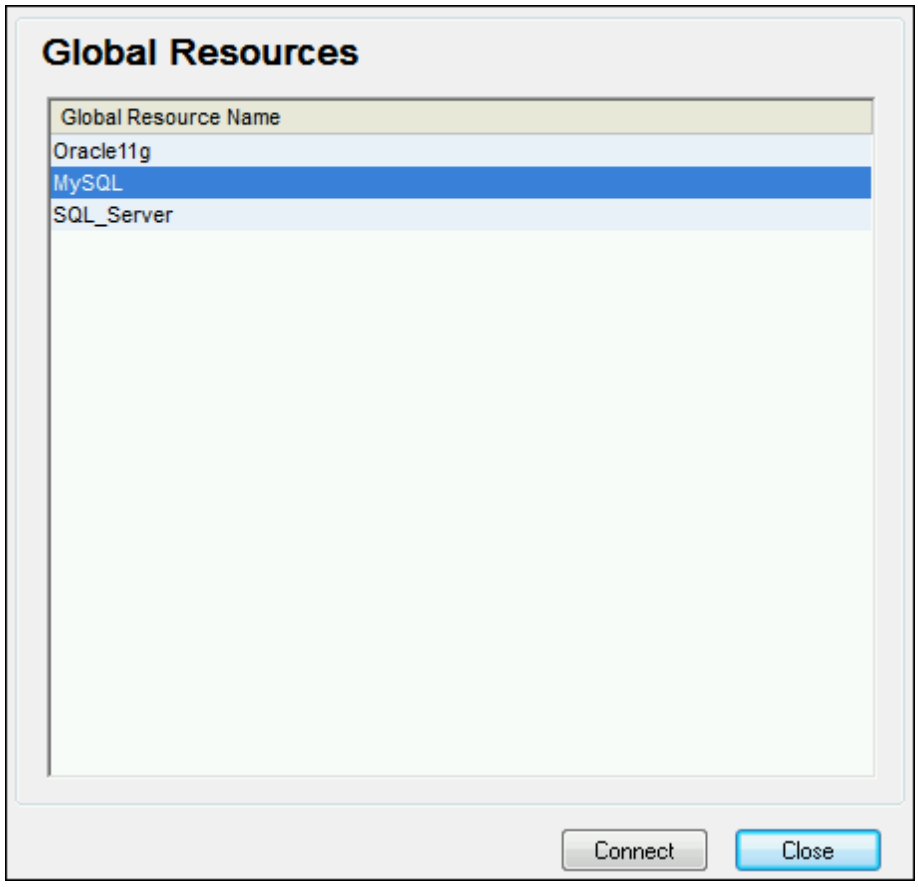
Using a Connection from Global Resources

Altova Global Resources represent a way to refer to files, folders, or databases so as to make these resources reusable, configurable and available across multiple Altova applications.

If you have already configured a database connection to be available as a global resource, you can reuse the connection at any time (even across different Altova applications).

To use a database connection from Global Resources:

1. Start the database connection wizard.
2. Click **Global Resources**. Any database connections previously configured as global resources are listed.



3. Select the database connection record, and click **Connect**.

Tip: To get additional information about each global resource, move the mouse cursor over the global resource.

Database Connection Examples

This section includes sample procedures for connecting to a database from MobileTogether Designer. Note that your Windows machine, the network environment, and the database client or server software is likely to have a configuration that is not exactly the same as the one presented in the following examples.

Note: 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.

Connecting to Firebird (ODBC)

This topic provides sample instructions for connecting 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 (http://www.firebirdsql.org/).
- 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 (http://www.firebirdsql.org/), 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.

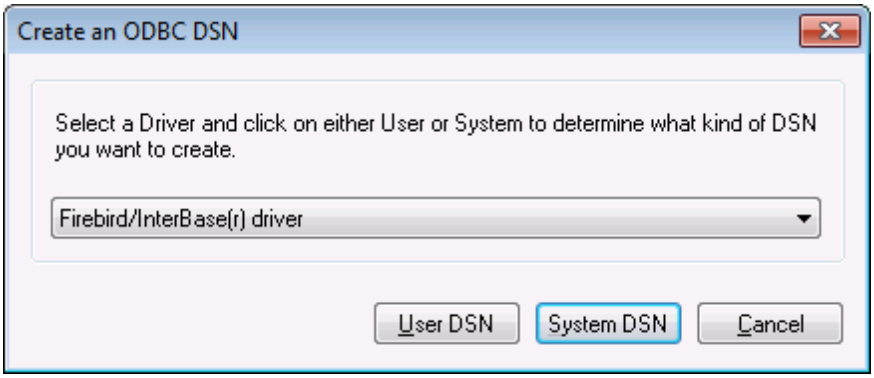
  Important:

  - The platform of both the Firebird ODBC driver and client (32-bit or 64-bit) must correspond to that of MobileTogether Designer.
  - The version of the Firebird client must correspond to the version of Firebird server to which you are connecting.

- You have the following database connection details: server host name or IP address, database path (or alias) on the server, user name, and password.

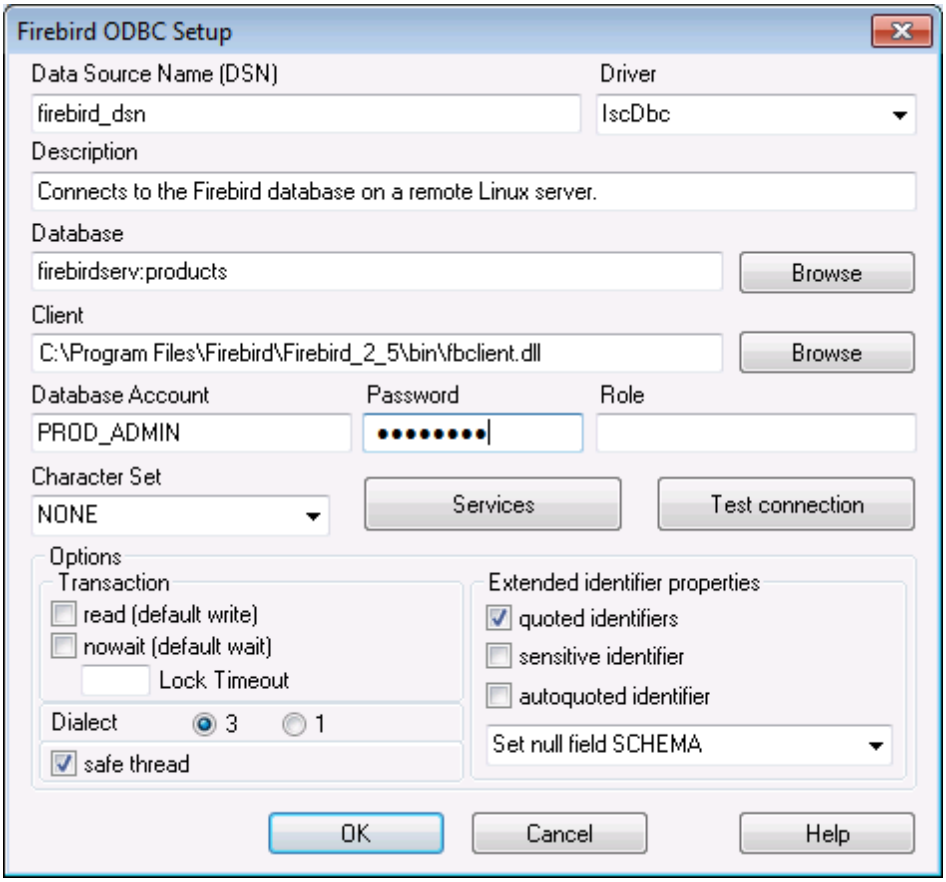
To connect to Firebird via ODBC:

1. Start the database connection wizard.
2. Click **ODBC Connections**.
3. Select **User DSN** (or **System DSN**, if you have administrative privileges), and then click **Add** .



4. 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).



5. 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:
	
	`firebirdserv:products`
	
	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).
	
	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:
	
	`firebirdserver:/var/Firebird/databases/`

	`butterflies.fdb`
	`127.0.0.1:D:\Misc\Lenders.fdb`
	
	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.

6. Click **OK**.

Connecting to Firebird (JDBC)

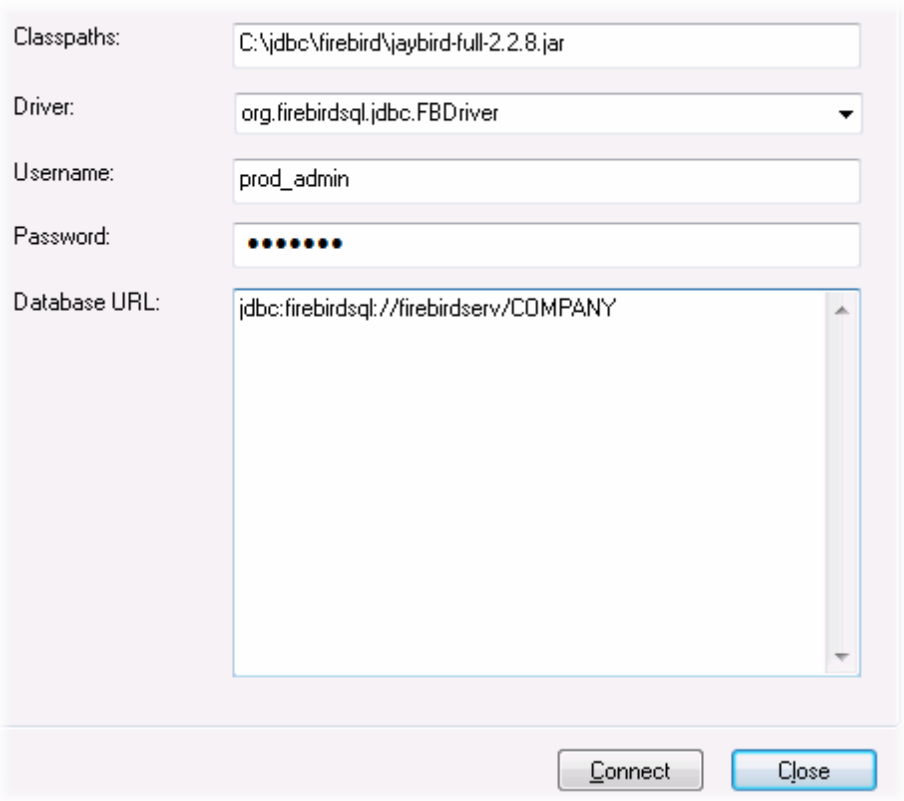
This topic provides sample instructions for connecting to a Firebird database server through JDBC.

Prerequisites:

- Java Runtime Environment (JRE) or Java Development Kit (JDK) must be installed on your operating system.
- 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 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 (http://www.firebirdsql.org/). This example uses *Jaybird 2.2.8*.
- You have the following database connection details: host, database path or alias, username, and password.

To connect to Firebird through JDBC:

1. Start the database connection wizard.
2. Click **JDBC Connections**.
3. Next to "Classpaths", enter the path to the .jar file which provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of .jar file paths. In this example, the required .jar file is located at the following path: **C:\jdbc\firebird\jaybird-full-2.2.8.jar**. Note that you can leave the "Classpaths" text box empty if you have added the .jar file path(s) to the CLASSPATH environment variable of the operating system (see also Configuring the CLASSPATH).
4. In the "Driver" box, select **org.firebirdsql.jdbc.FBDriver**. Note that this entry is available if a valid .jar file path is found either in the "Classpath" text box, or in the operating system's CLASSPATH environment variable (see the previous step).



5. Enter the username and password to the database in the corresponding text boxes.
6. Enter the connection string to the database server in the Database URL text box, by replacing the highlighted values with the ones applicable to your database server.

```
jdbc:firebirdsql://<host>[:<port>]/<database path or alias>
```

7. Click **Connect**.

Connecting to IBM DB2 (ODBC)

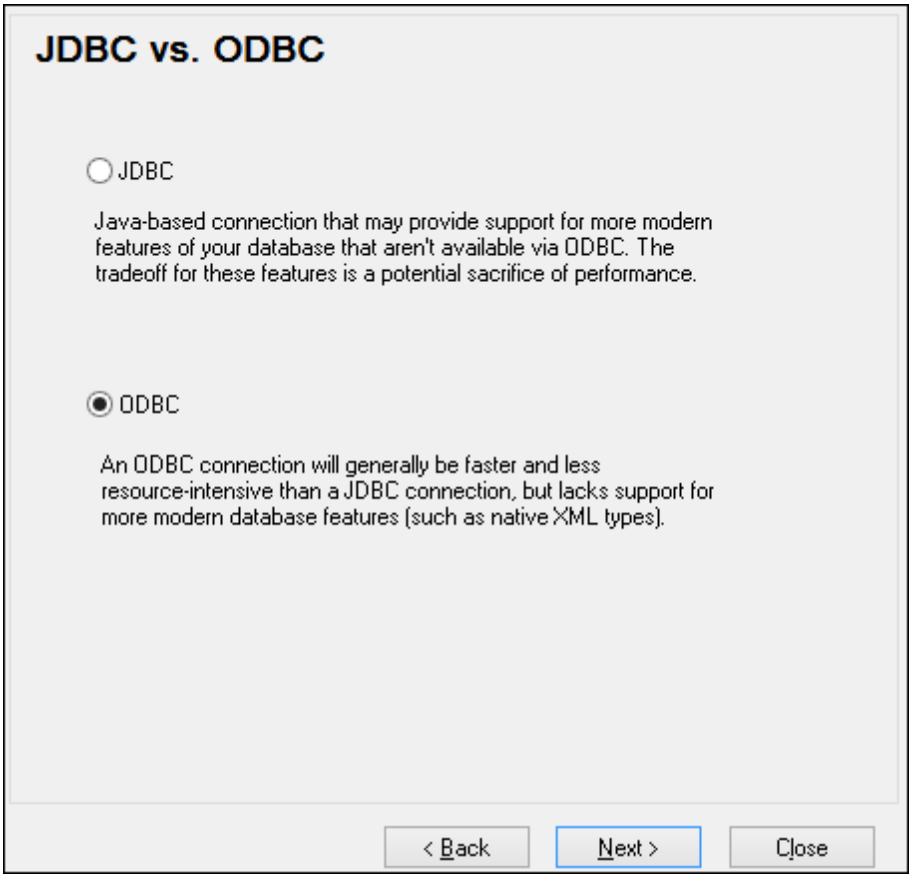
This topic provides sample instructions for connecting to an IBM DB2 database through ODBC.

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).
- 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.

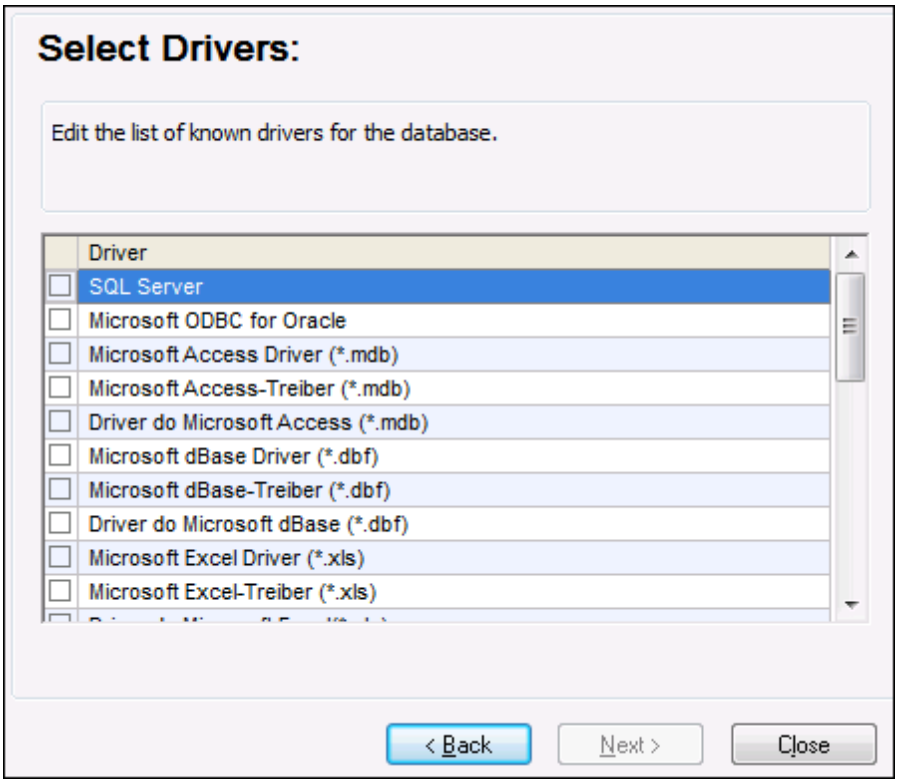
To connect to IBM DB2:

1. Start the database connection wizard and select **IBM DB2 (ODBC/JDBC)**.
2. Click **Next**.

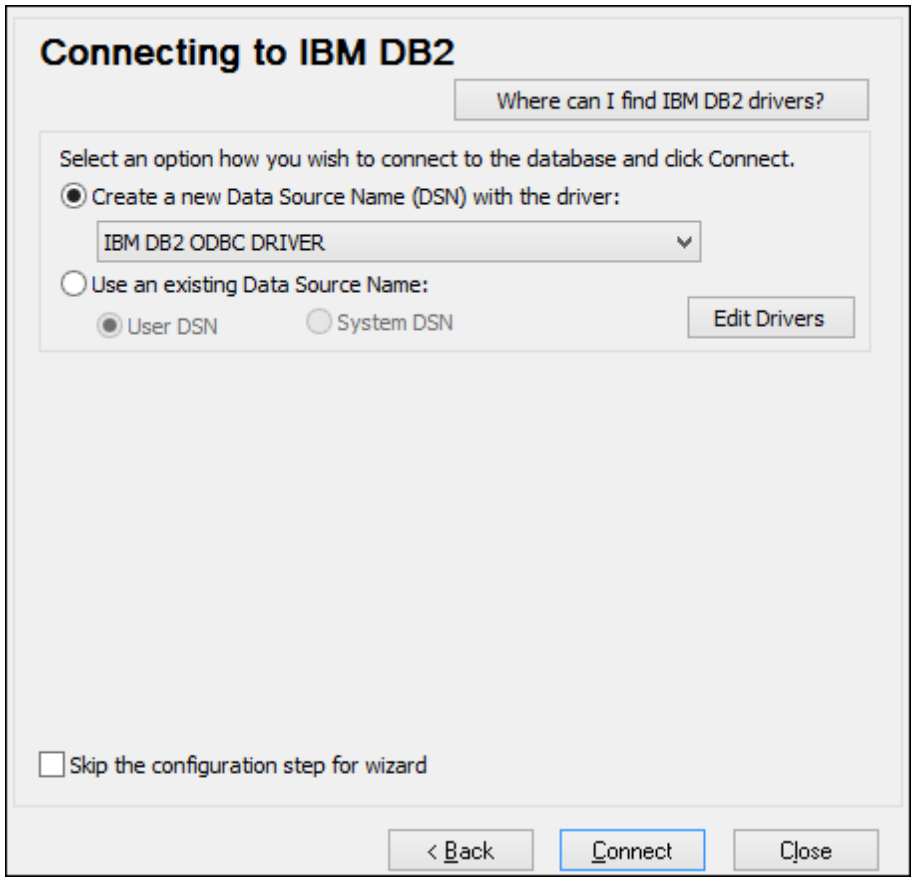


3. Select **ODBC**, and click **Next**. If prompted to edit the list of known drivers for the database, select the database drivers applicable to IBM DB2 (see Prerequisites), and

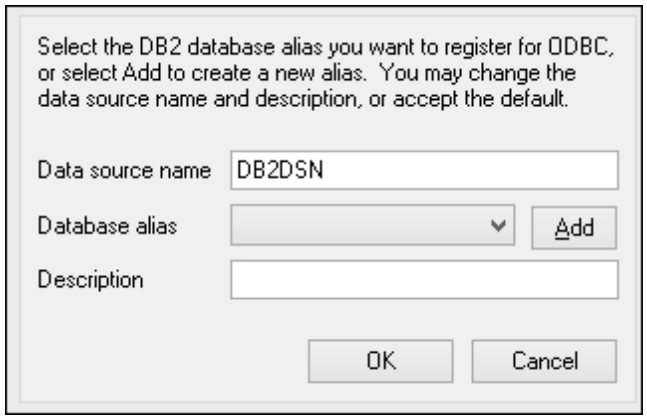
click **Next**.



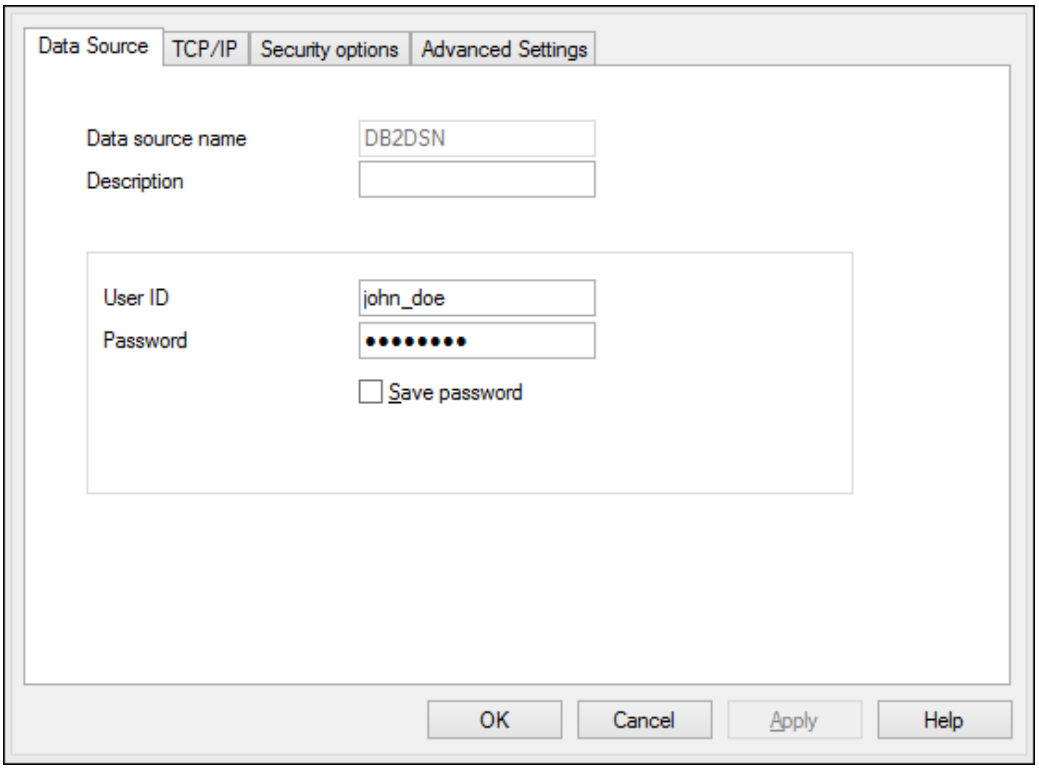
4. Select the IBM DB2 driver from the list, and then click **Connect**. (To edit the list of available drivers, click **Edit Drivers**, and then check or uncheck the IBM DB2 drivers you wish to add or remove, respectively.)



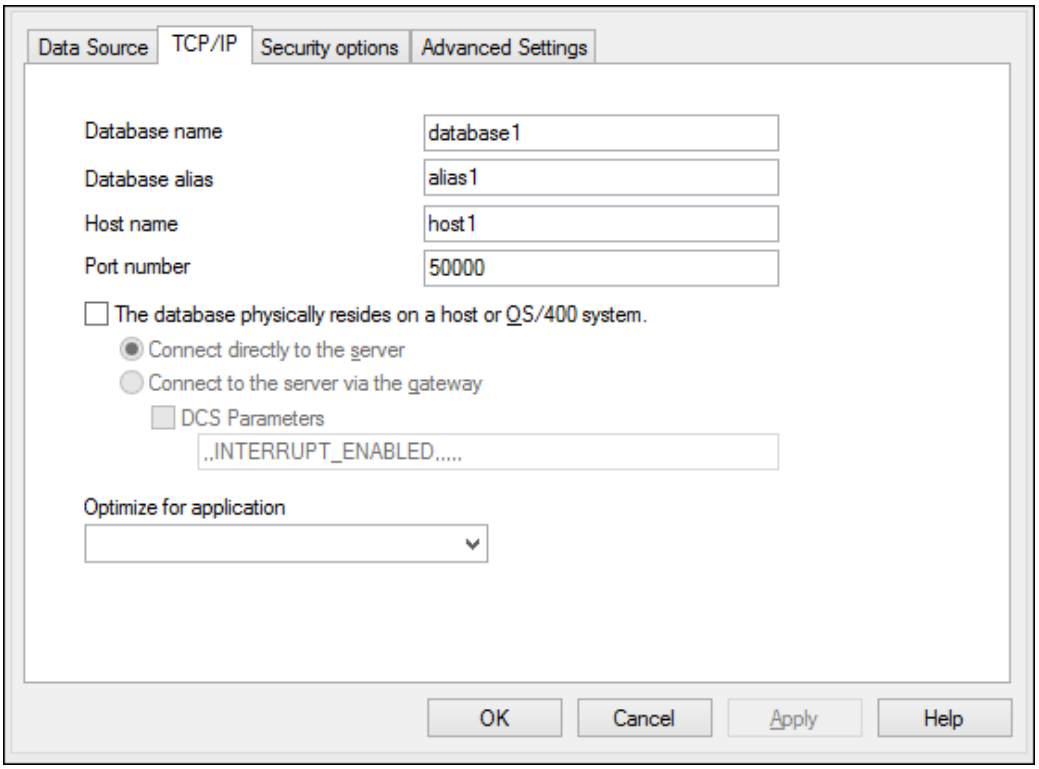
5. Enter a data source name (in this example, **DB2DSN**), and then click **Add**.



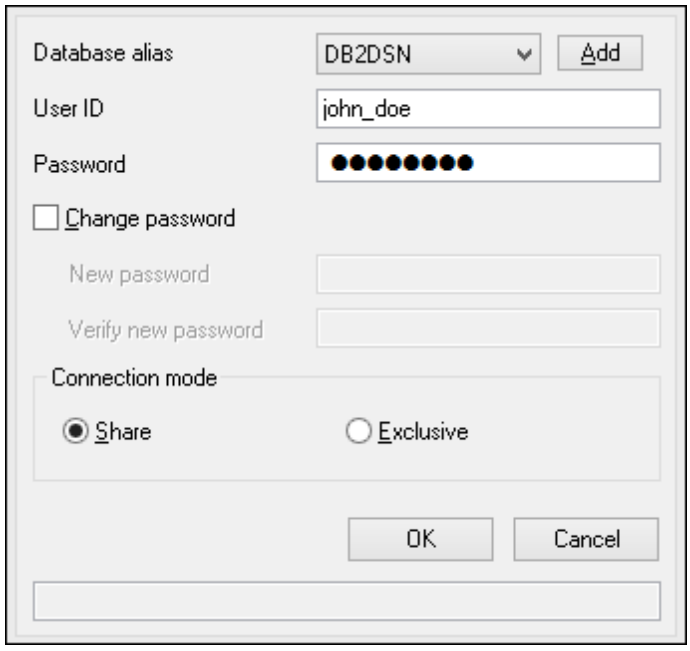
6. On the **Data Source** tab, enter the user name and password to the database.



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.



8. Enter again the username and password, and then click **OK**.

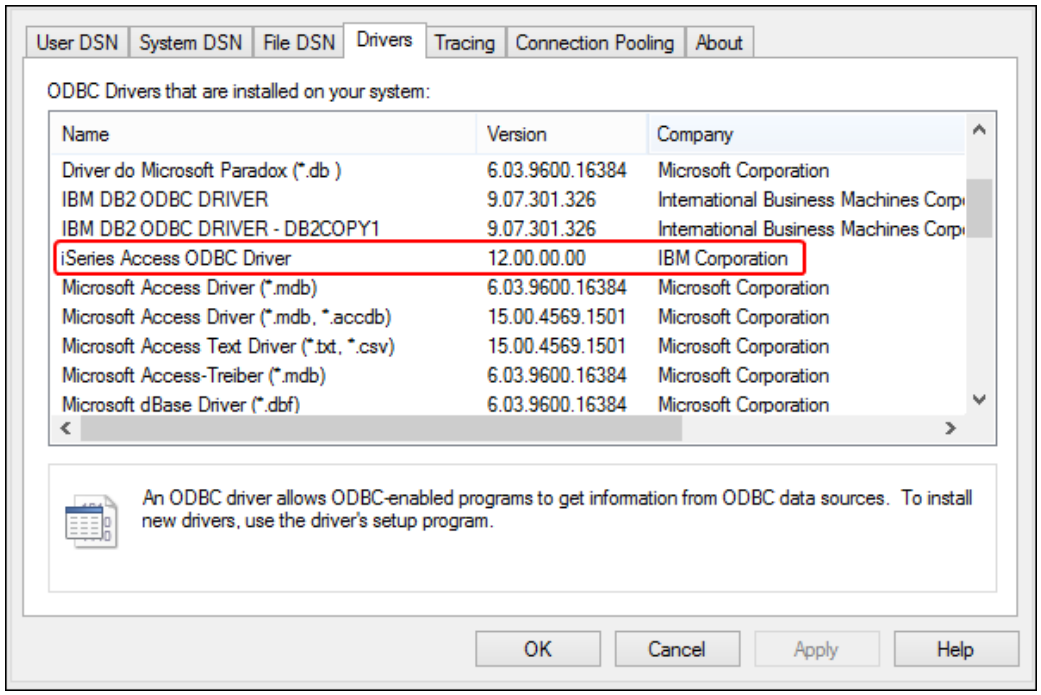


Connecting to IBM DB2 for i (ODBC)

This topic provides sample instructions for connecting to an *IBM DB2 for i* database through ODBC.

Prerequisites:

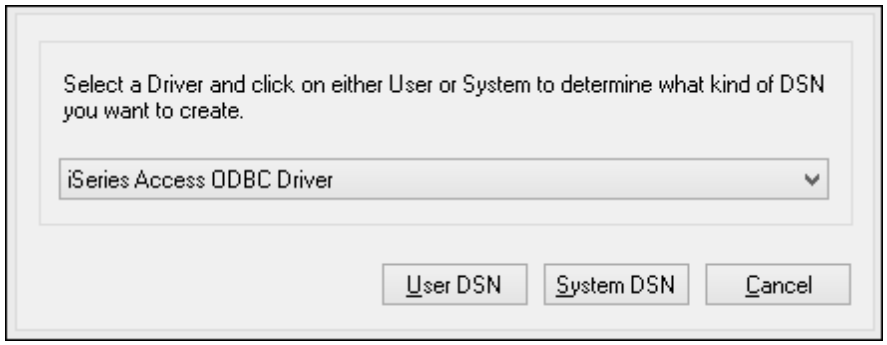
- *IBM System i Access for Windows* must be installed on your operating system (this example uses *IBM System i Access for Windows V6R1M0*). For installation instructions, check the documentation supplied with your *IBM DB2 for i* software. After installation, check if the ODBC driver is available on your machine (see Viewing the Available ODBC Drivers).



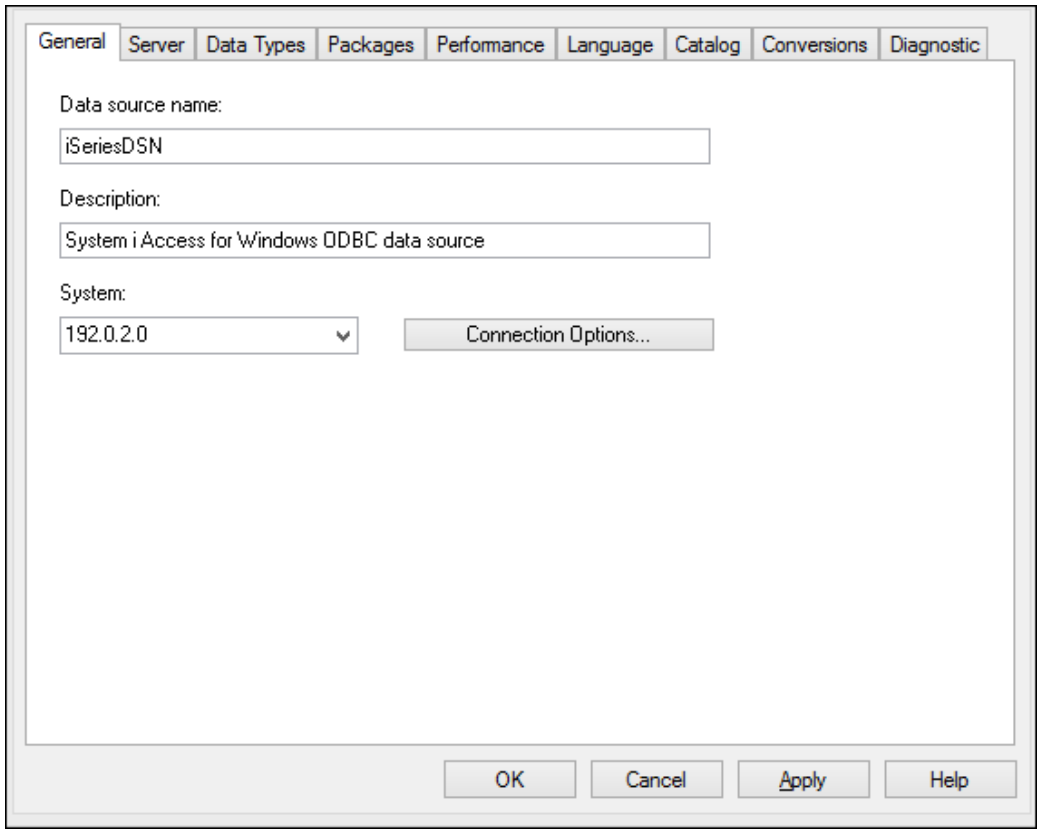
- You have the following database connection details: the I.P. address of the database server, database user name, and password.
- Run *System i Navigator* and follow the wizard to create a new connection. When prompted to specify a system, enter the I.P. address of the database server. After creating the connection, it is recommended to verify it (click on the connection, and select **File** > **Diagnostics** > **Verify Connection**). If you get connectivity errors, contact the database server administrator.

To connect to IBM DB2 for i:

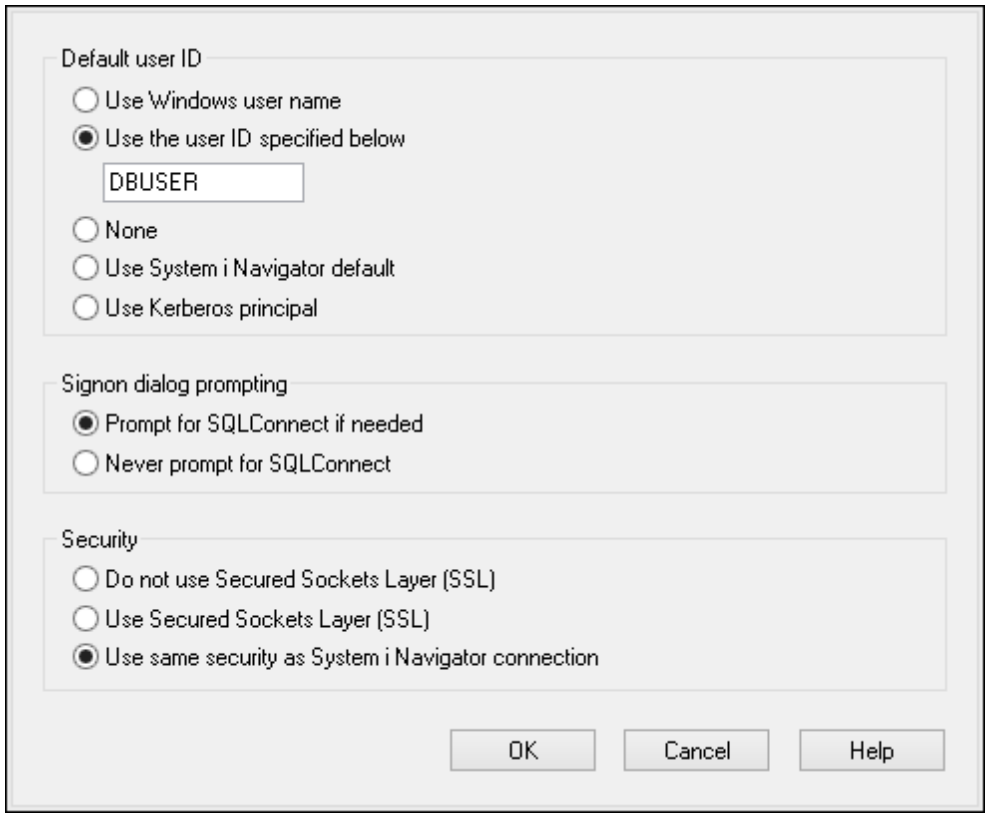
1. Start the database connection wizard.
2. Click **ODBC connections**.
3. Click **User DSN** (alternatively, click **System DSN**, or **File DSN**, in which case the subsequent instructions will be similar).
4. Click **Add** .
5. Select the **iSeries Access ODBC Driver** from the list, and click **User DSN** (or **System DSN**, if applicable).



6. 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**.



7. Click Connection Options, select **Use the User ID specified below** and enter the name of the database user (in this example, **DBUSER**).



8. Click **OK**. The new data source becomes available in the list of DSNs.
9. Click **Connect**.
10. Enter the user name and password to the database when prompted, and then click **OK**.

Connecting to IBM Informix (JDBC)

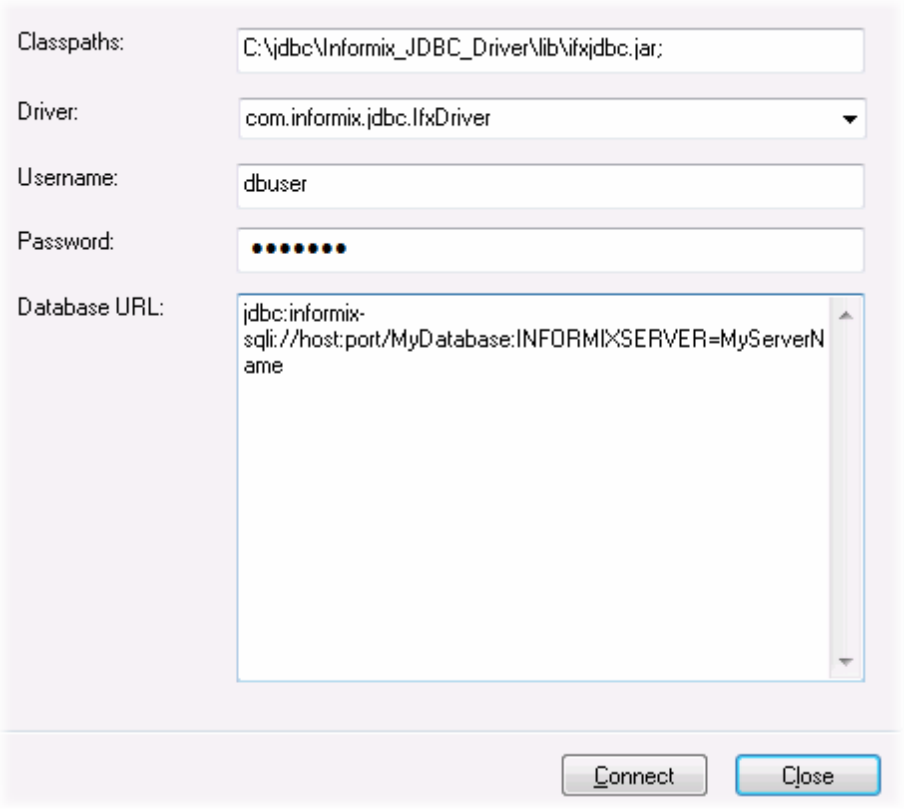
This topic provides sample instructions for connecting to an IBM Informix database server through JDBC.

Prerequisites:

- Java Runtime Environment (JRE) must be installed on your operating system.
- 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.

To connect to IBM Informix through JDBC:

1. Start the database connection wizard.
2. Click **JDBC Connections**.
3. Next to "Classpaths", enter the path to the .jar file which provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of .jar file paths. In this example, the required .jar file is located at the following path: **C:\Informix_JDBC_Driver\lib\ifxjdbc.jar**. Note that you can leave the "Classpaths" text box empty if you have added the .jar file path(s) to the CLASSPATH environment variable of the operating system (see also Configuring the CLASSPATH).
4. In the "Driver" box, select **com.informix.jdbc.IfxDriver**. Note that this entry is available if a valid .jar file path is found either in the "Classpath" text box, or in the operating system's CLASSPATH environment variable (see the previous step).



5. Enter the username and password to the database in the corresponding text boxes.
6. Enter the connection string to the database server in the Database URL text box, by replacing the highlighted values with the ones applicable to your database server.

```
jdbc:informix-sqli://hostName:port/
databaseName:INFORMIXSERVER=myserver;
```

7. Click **Connect**.

Connecting to MariaDB (ODBC)

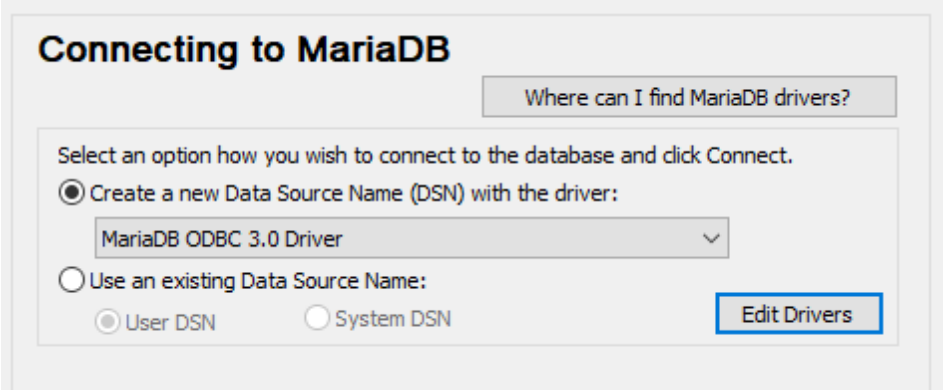
This example illustrates how to connect to a MariaDB database server through ODBC.

Prerequisites:

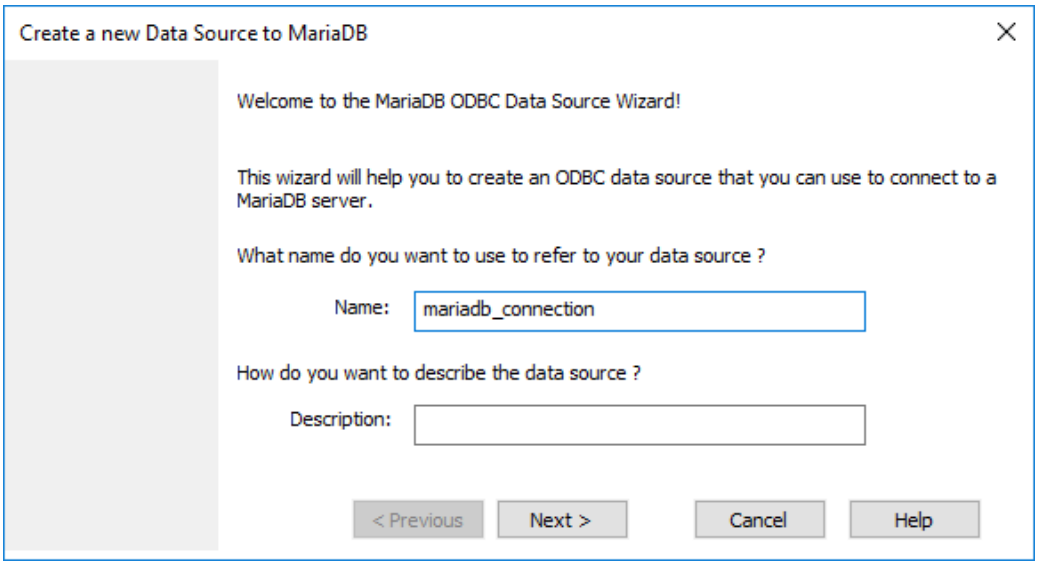
- The MariaDB Connector/ODBC (https://downloads.mariadb.org/connector-odbc/) must be installed.
- You have the following database connection details: host, database, port, username, and password.

To connect to MariaDB through ODBC:

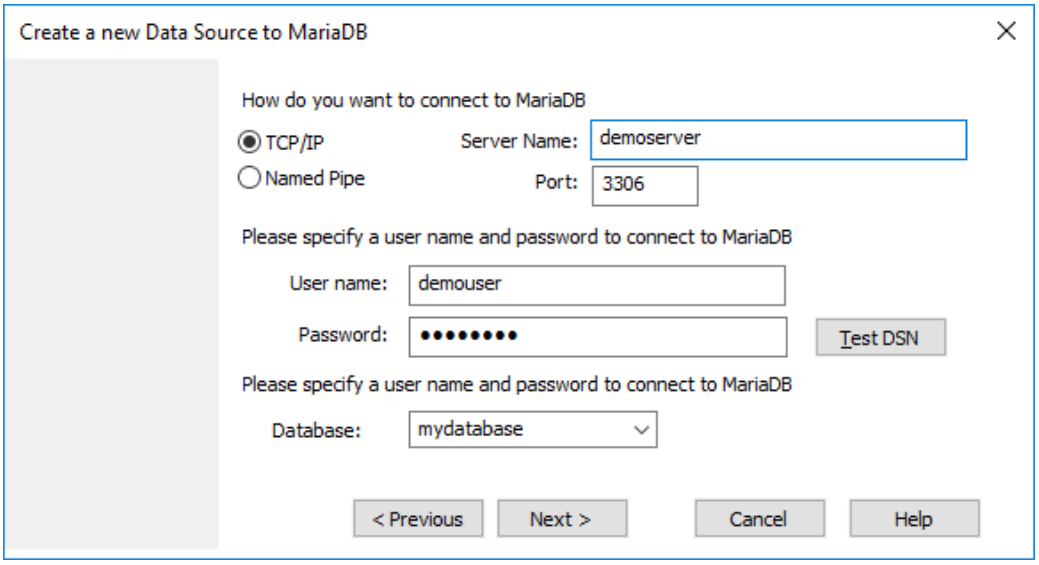
1. Start the database connection wizard.
2. Select **MariaDB (ODBC)**, and then click **Next**.



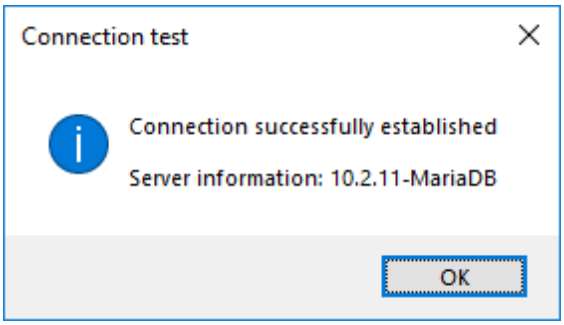
3. Select **Create a new Data Source Name (DSN) with the driver**, and choose **MariaDB ODBC 3.0 Driver**. If no such driver is available in the list, click **Edit Drivers**, and select any available MariaDB drivers (the list contains all ODBC drivers installed on your operating system).
4. Click **Connect**.



5. Enter name and, optionally, a description that will help you identify this ODBC data source in future.



6. Fill in the database connection credentials (TCP/IP Server, User, Password), select a database, and then click **Test DSN**. Upon successful connection, a message box appears:



7. Click **Next** and complete the wizard. Other parameters may be required, depending on the case (for example, SSL certificates if you are connecting to MariaDB through 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.

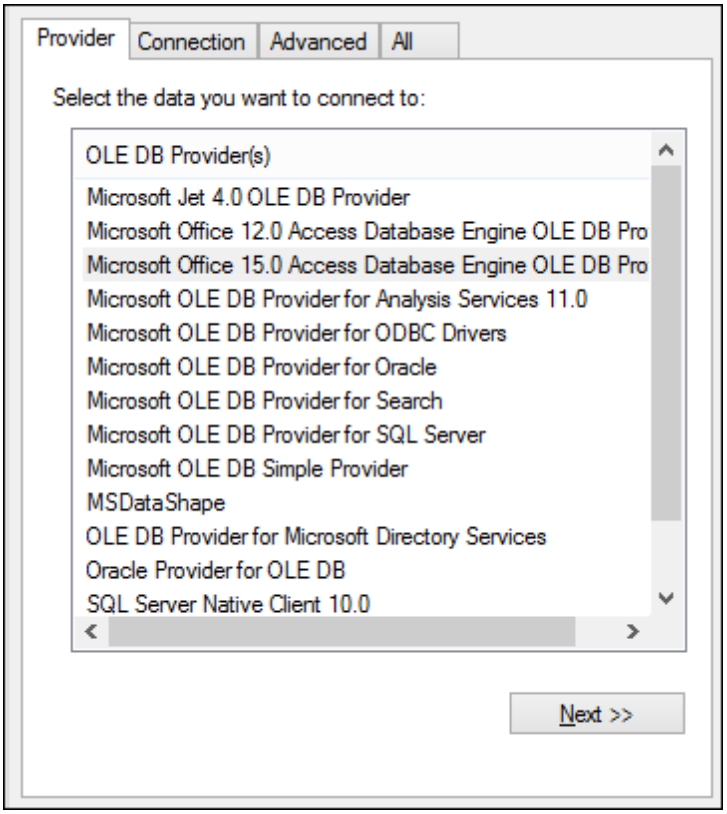
Connecting to Microsoft Access (ADO)

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. 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.

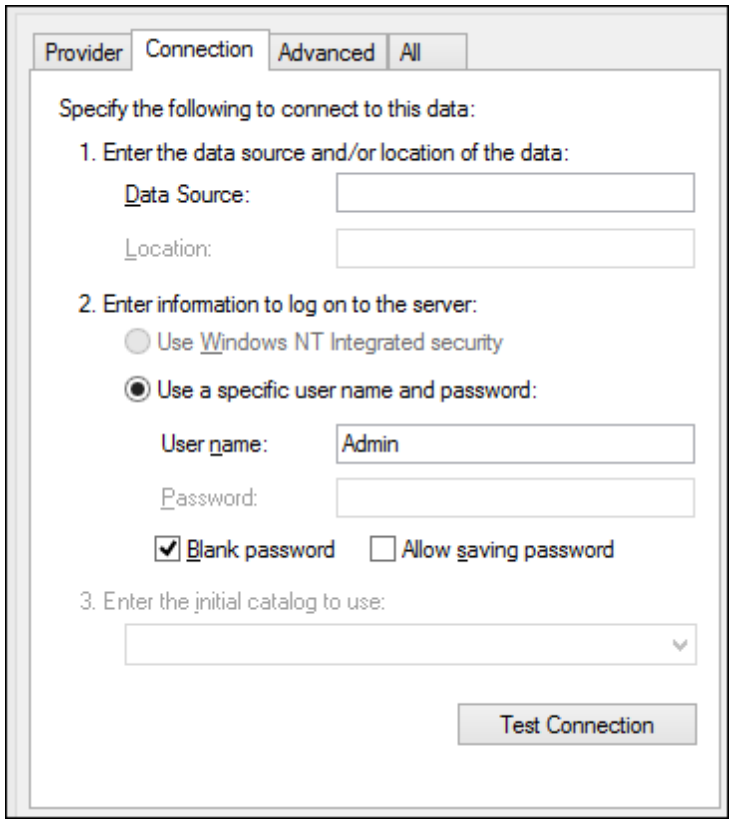
It is also possible to connect to Microsoft Access through an ODBC connection, but there are some limitations in this scenario, so it is best to avoid it.

To connect to a password-protected Microsoft Access database:

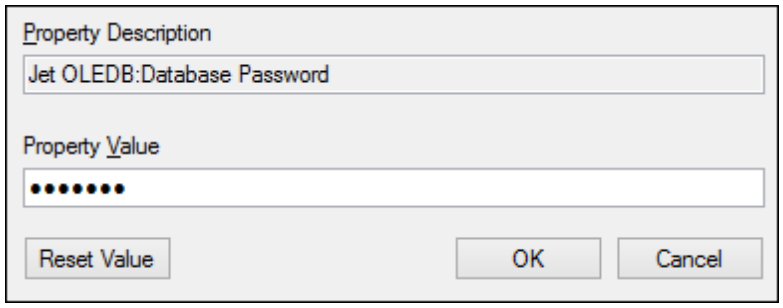
1. Start the database connection wizard.
2. Click **ADO Connections**.
3. Click **Build**.



4. Select the **Microsoft Office 15.0 Access Database Engine OLE DB Provider**, and then click **Next**.



5. In the Data Source box, enter the path to the Microsoft Access file. Because the file is on the local network share **U:\Departments\Finance\Reports\Revenue.accdb**, we will convert it to UNC format, and namely **\\server1\\dfs\Departments\Finance\Reports\Revenue.accdb**, where **server1** is the name of the server and **dfs** is the name of the network share.
6. On the **All** tab, double click the **Jet OLEDB:Database Password** property and enter the database password as property value.



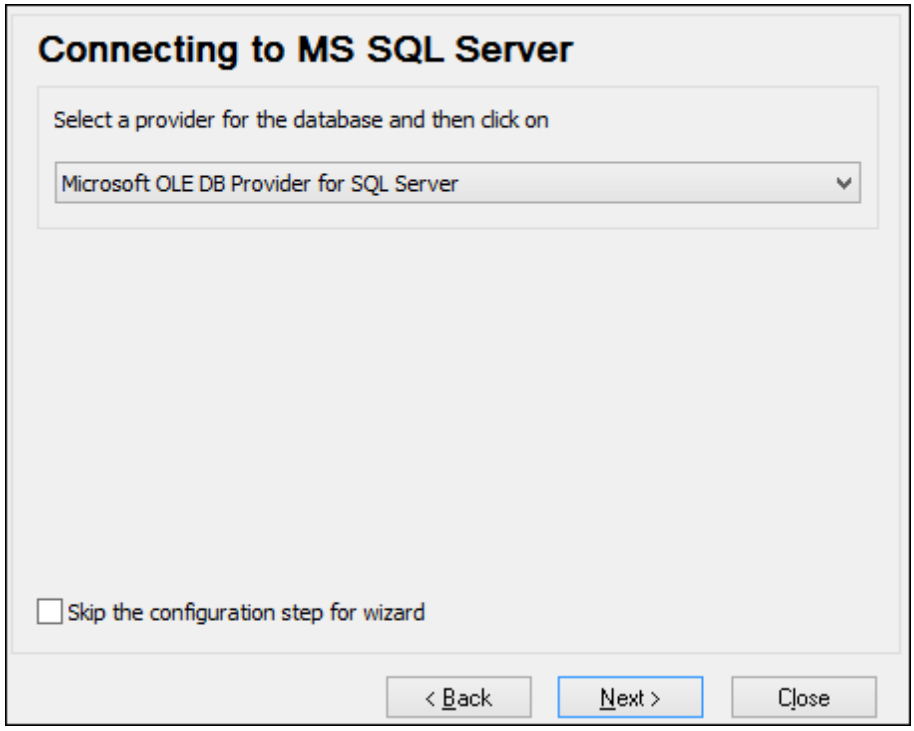
Note: If you are still unable to connect, locate the workgroup information file (**System.MDW**) applicable to your user profile (see http://support.microsoft.com/kb/305542 for instructions), and set the value of the **Jet OLEDB: System database** property to the path of the **System.MDW** file.

Connecting to Microsoft SQL Server (ADO)

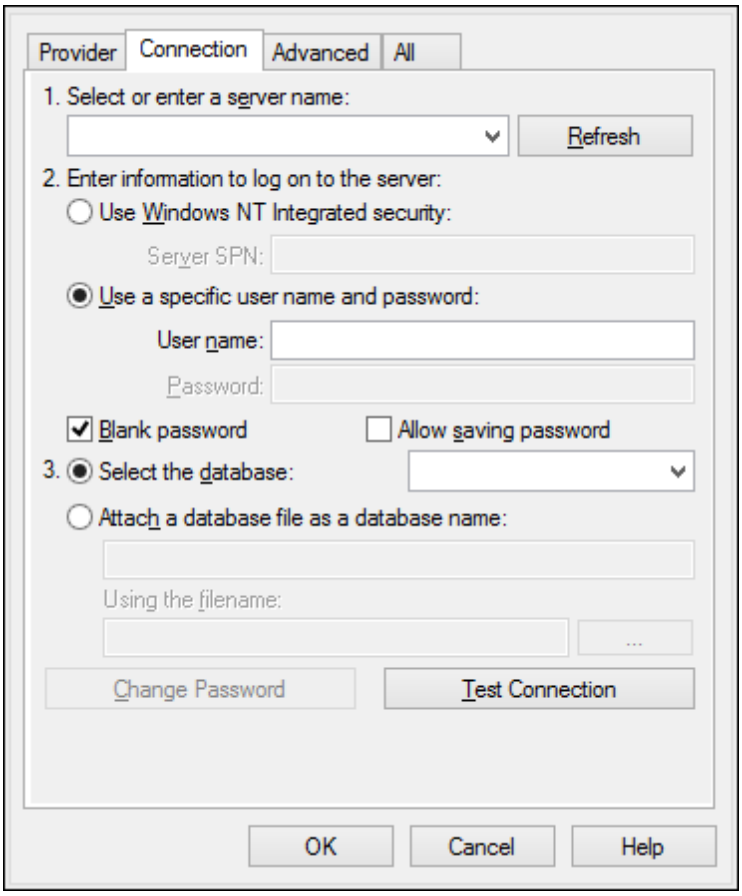
This example illustrates how to connect to a SQL Server database through ADO.

To connect to SQL Server using the Microsoft OLE DB Provider:

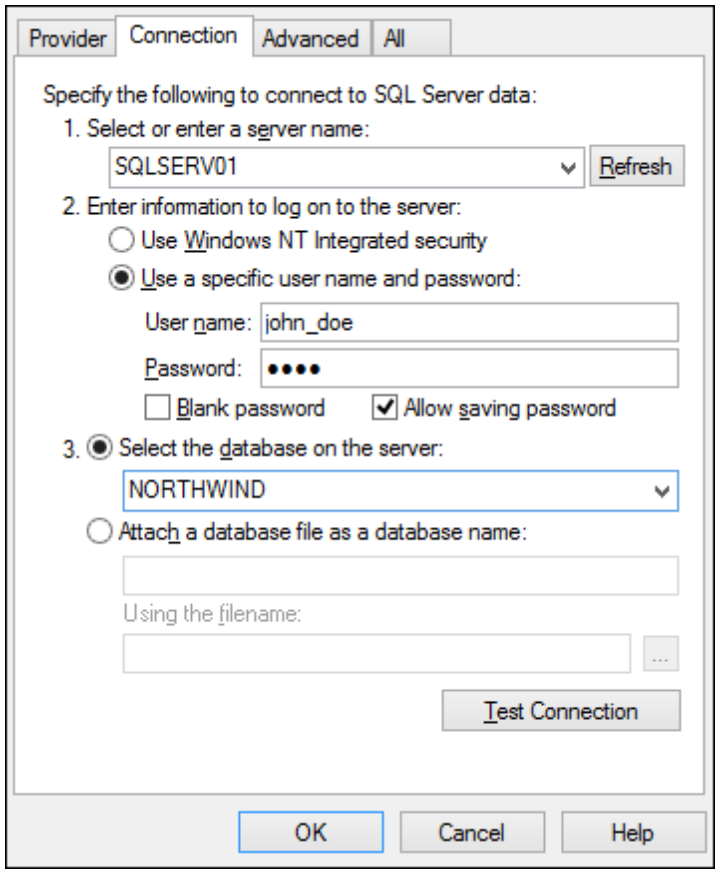
1. Start the database connection wizard.
2. Select **Microsoft SQL Server (ADO)**, and then click **Next**. The list of available ADO drivers is displayed.



3. Select **Microsoft OLE DB Provider for SQL Server**, and then click **Next**.



4. Select or enter the name of the database server (in this example, **SQLSERV01**). To view the list of all servers on the network, expand the drop-down list.
5. If the database server was configured to allow connections from users authenticated on the Windows domain, select **Use Windows NT integrated security**. Otherwise, select **Use a specific user name and password**, and type them in the relevant boxes.
6. Select the database to which you are connecting (in this example, **NORTHWIND**).
7. To test the connection at this time, click **Test Connection**. This is an optional, recommended step.
8. Do one of the following:
   a. Select the **Allow saving password** check box.
   b. On the **All** tab, change the value of the **Persist Security Info** property to **True**.


Provider
Connection
Advanced
All
Specify the following to connect to SQL Server data:
1. Select or enter a server name:
SQLSERV01
Refresh
2. Enter information to log on to the server:
Use Windows NT Integrated security
Use a specific user name and password:
User name: john_doe
Password: ••••
Blank password
Allow saving password
3. Select the database on the server:
NORTHWIND
Attach a database file as a database name:
Using the filename:
Test Connection
OK
Cancel
Help

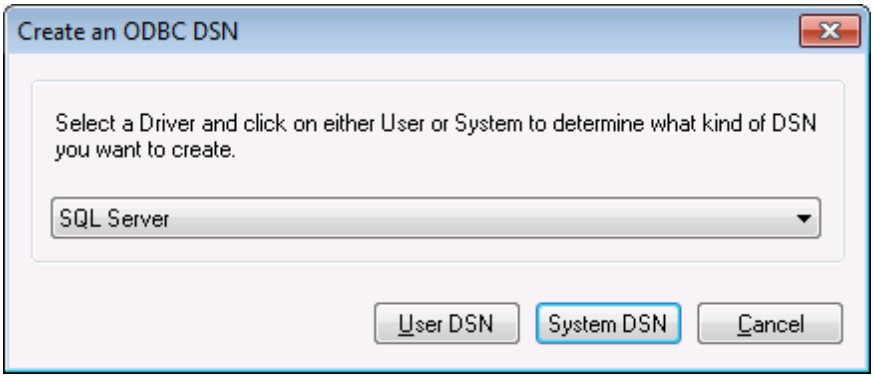
9. Click **OK**.

Connecting to Microsoft SQL Server (ODBC)

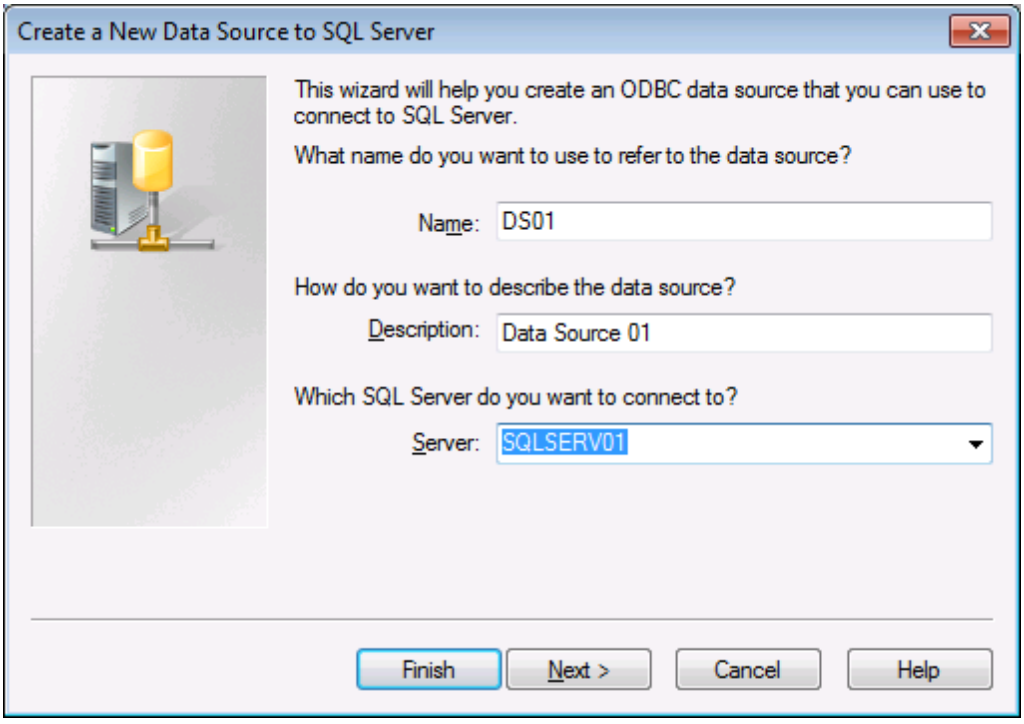
This example illustrates how to connect to a SQL Server database through ODBC.

To connect to SQL Server using ODBC:

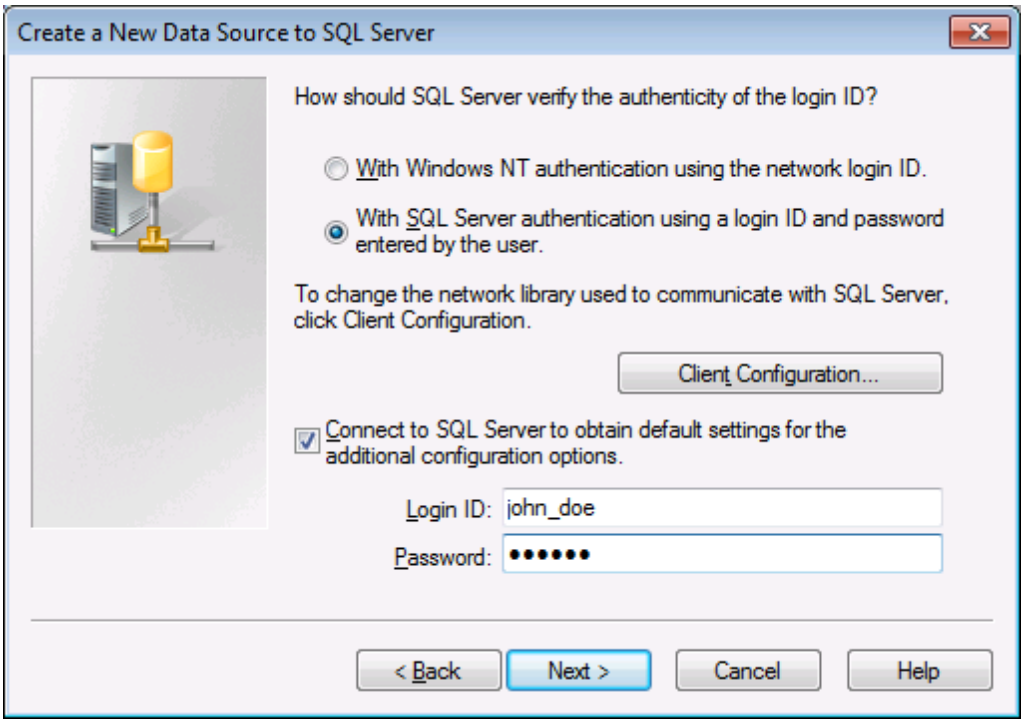
1. Start the database connection wizard.
2. Click **ODBC Connections**.
3. Select **User DSN** (or **System DSN**, if you have administrative privileges), and then click **Add** .



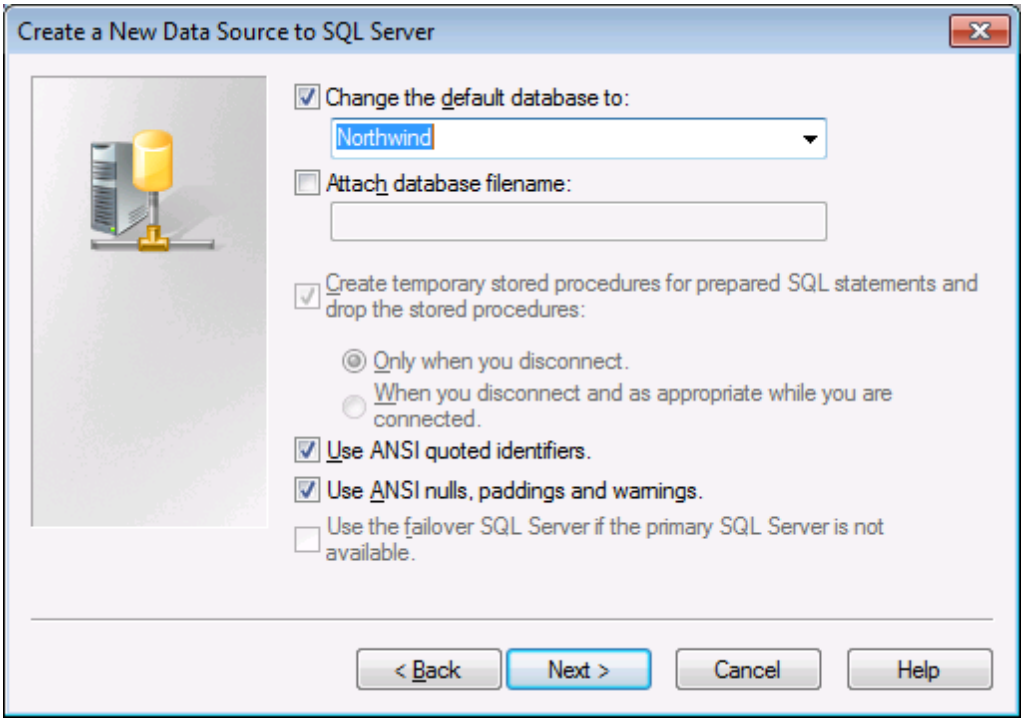
4. Select **SQL Server** (or **SQL Server Native Client**, if available), and then click **User DSN** (or **System DSN** if you are creating a System DSN).



5. Enter a name and description to identify this connection, and then select from the list the SQL Server to which you are connecting (**SQLSERV01** in this example).



6. If the database server was configured to allow connections from users authenticated on the Windows domain, select **With Windows NT authentication**. Otherwise, select **With SQL Server authentication...** and type the user name and password in the relevant boxes.



7. Select the name of the database to which you are connecting (in this example,

Northwind).
8. Click **Finish**.

Connecting to MySQL (ODBC)

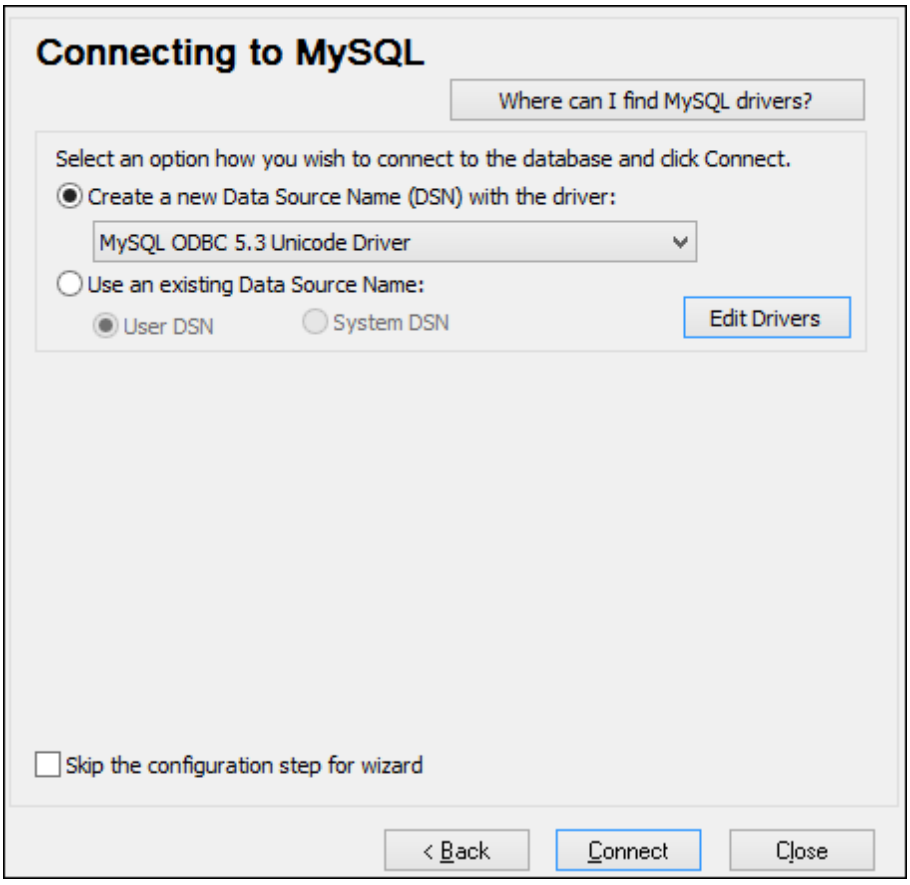
This topic provides sample instructions for connecting to a MySQL database server from a Windows machine through the ODBC driver. The MySQL ODBC driver is not available on Windows, so it must be downloaded and installed separately. This example uses MySQL ODBC driver version 5.3.4 downloaded from the official website (see also Database Drivers Overview).

Prerequisites:

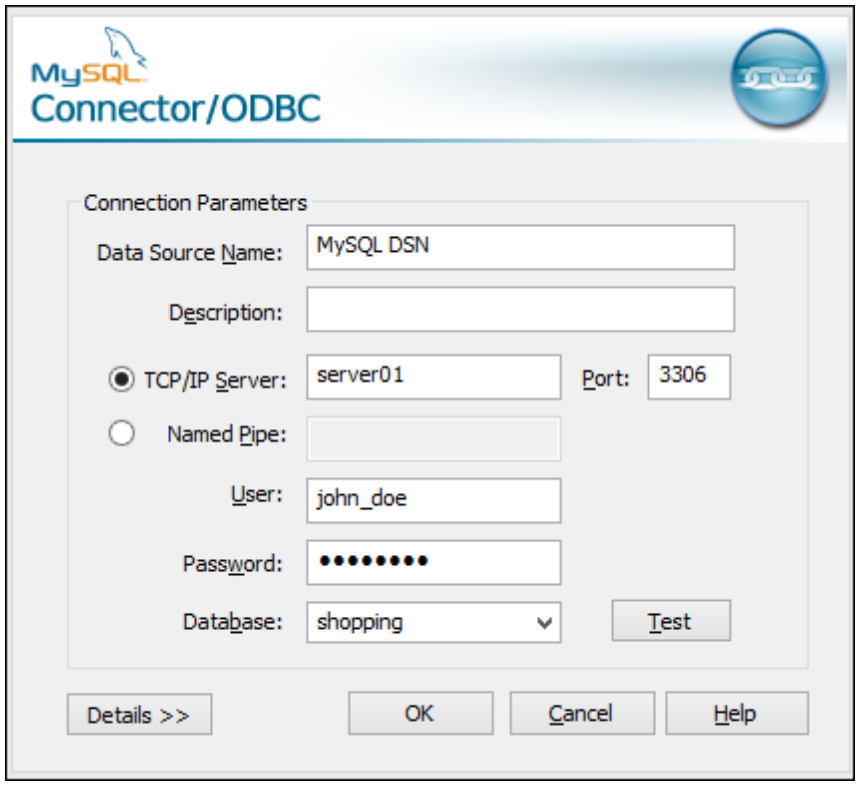
- MySQL ODBC driver must be installed on your operating system (for installation instructions, check the documentation supplied with the driver).
- You have the following database connection details: host, database, port, username, and password.

To connect to MySQL via ODBC:

1. Start the database connection wizard.
2. Select **MySQL (ODBC)**, and then click **Next**.



3. Select **Create a new Data Source Name (DSN) with the driver**, and select a MySQL driver. If no MySQL driver is available in the list, click **Edit Drivers**, and select any available MySQL drivers (the list contains all ODBC drivers installed on your operating system).
4. Click **Connect**.



5. In the Data Source Name box, enter a descriptive name that will help you identify this ODBC data source in future.
6. 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 their default values.

Connecting to Oracle (ODBC)

This example illustrates a common scenario where you connect from MobileTogether Designer to an Oracle database server on a network machine, through 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 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.

Prerequisites:

- The Oracle database client (which includes the ODBC Oracle driver) must be installed and configured on your operating system. For instructions on how to install and configure an Oracle database client, refer to the documentation supplied with your Oracle software.
- 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)
    )
  )
```

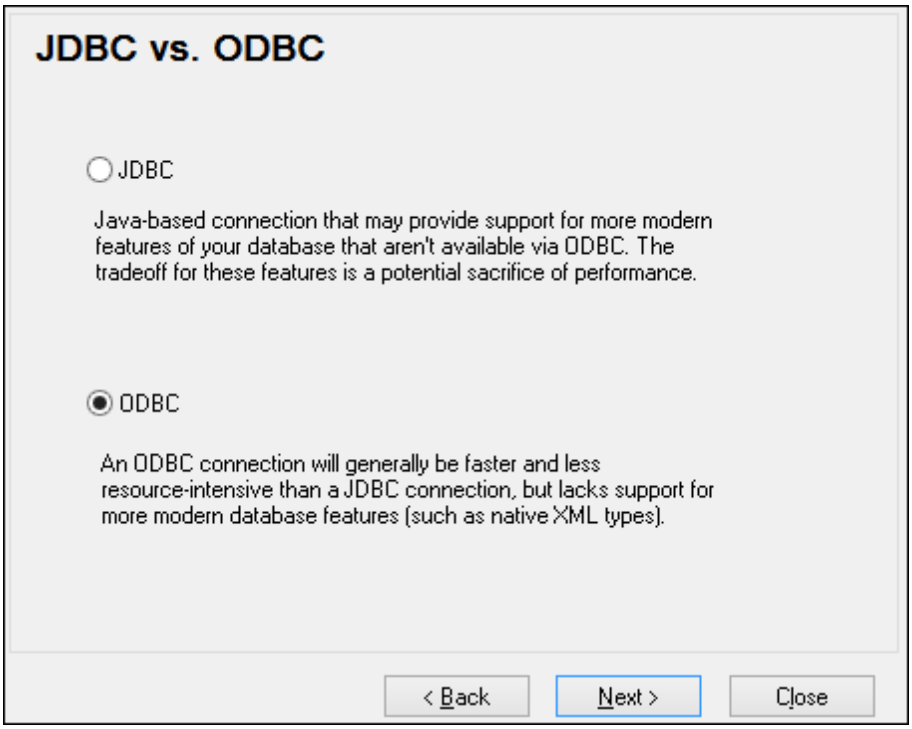
The path to the **tnsnames.ora** file depends on the location where Oracle home directory was installed. For Oracle database client 11.2.0, the default Oracle home directory path could be as follows:

```
C:\app\username\product\11.2.0\client_1\network\admin\tnsnames.ora
```

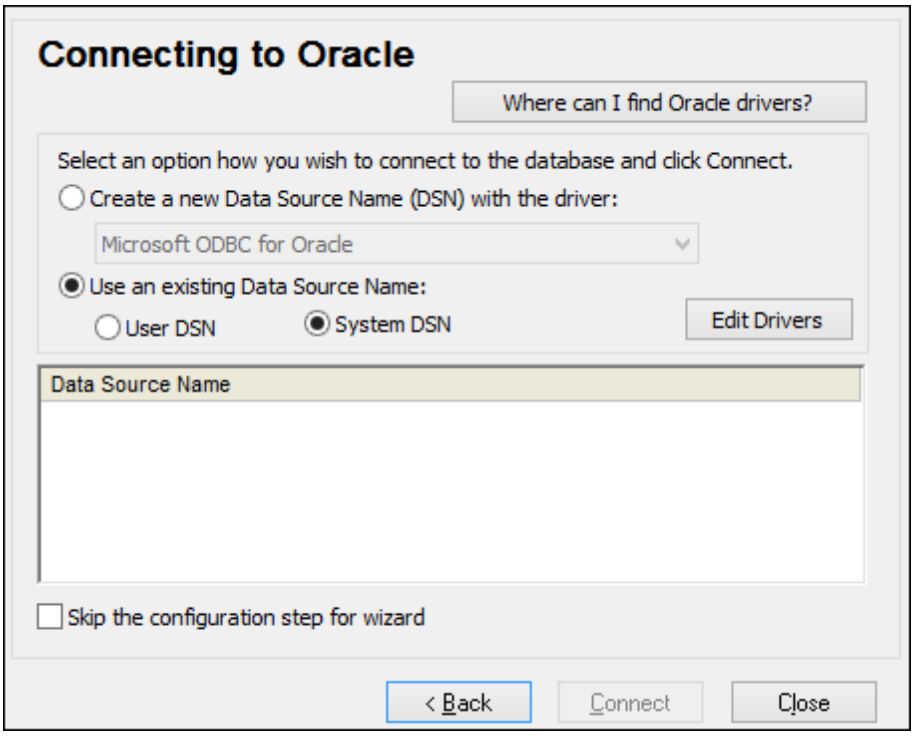
You can add new entries to the **tnsnames.ora** file either by pasting the connection details and saving the file, or by running the Oracle *Net Configuration Assistant* wizard (if available).

To connect to Oracle using ODBC:

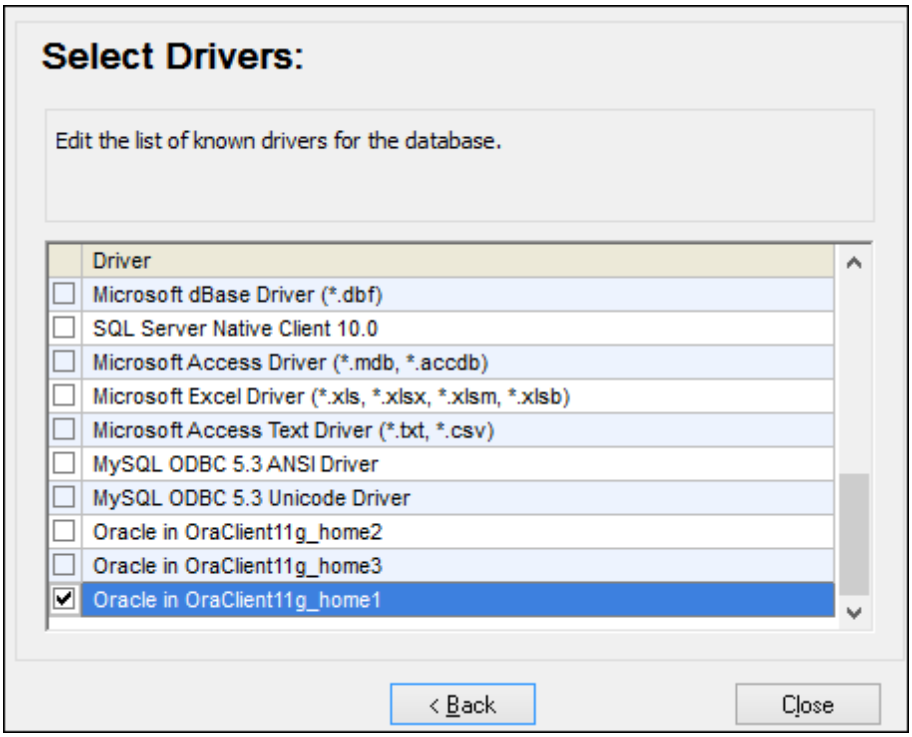
1. Start the database connection wizard.
2. Select **Oracle (ODBC / JDBC)**, and then click **Next**.



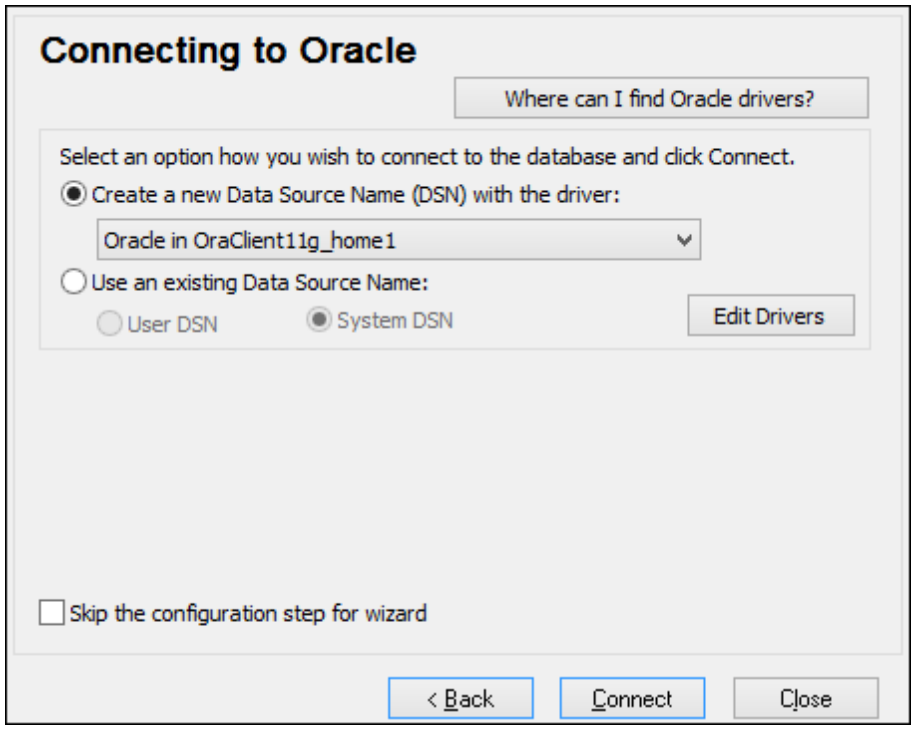
3. Select **ODBC**.



4. Click **Edit Drivers**.

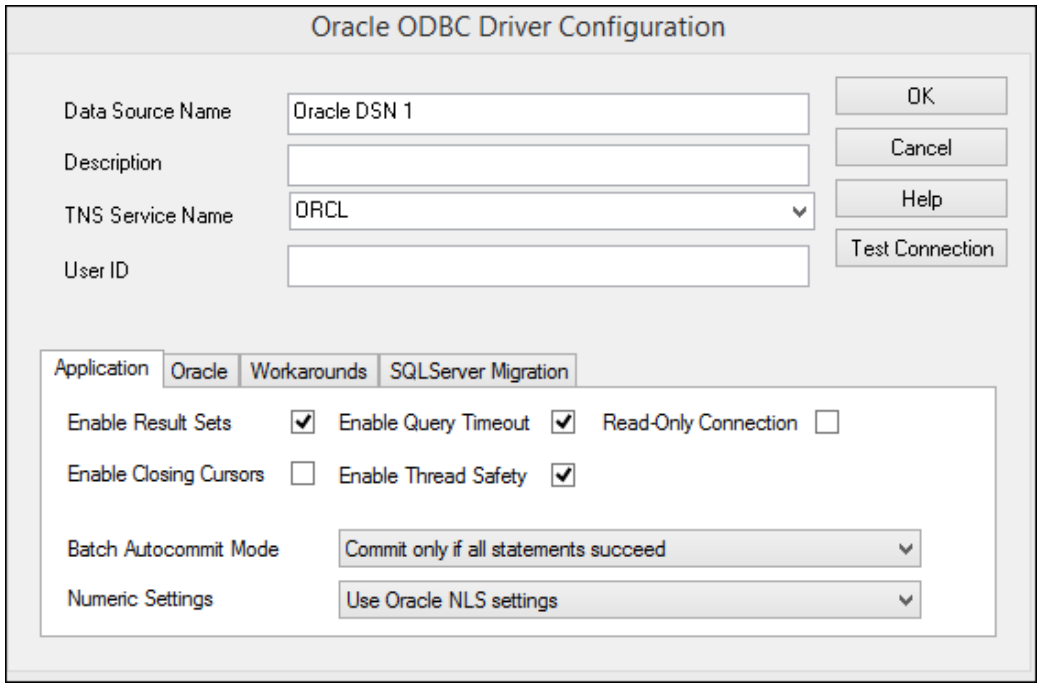


5. Select the Oracle drivers you wish to use (in this example, **Oracle in OraClient11g_home1**). The list displays the Oracle drivers available on your system after installation of Oracle client.
6. Click **Back**.
7. Select **Create a new data source name (DSN) with the driver**, and then select the Oracle driver chosen in step 4.

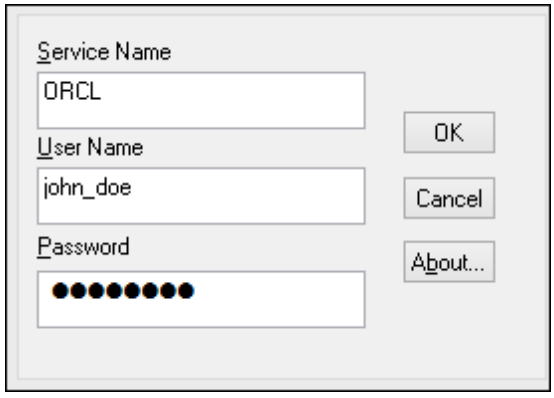


Avoid using the Microsoft-supplied driver called **Microsoft ODBC for Oracle** driver. Microsoft recommends using the ODBC driver provided by Oracle (see http://msdn.microsoft.com/en-us/library/ms714756%28v=vs.85%29.aspx)

8. Click **Connect**.



9. In the Data Source Name text box, enter a name to identify the data source (in this example, **Oracle DSN 1**).
10. In the TNS Service Name box, enter the connection name as it is defined in the **tnsnames.ora** file (see prerequisites). In this example, the connection name is **ORCL**.
11. Click **OK**.



12. Enter the username and password to the database, and then click OK.

Connecting to Oracle (JDBC)

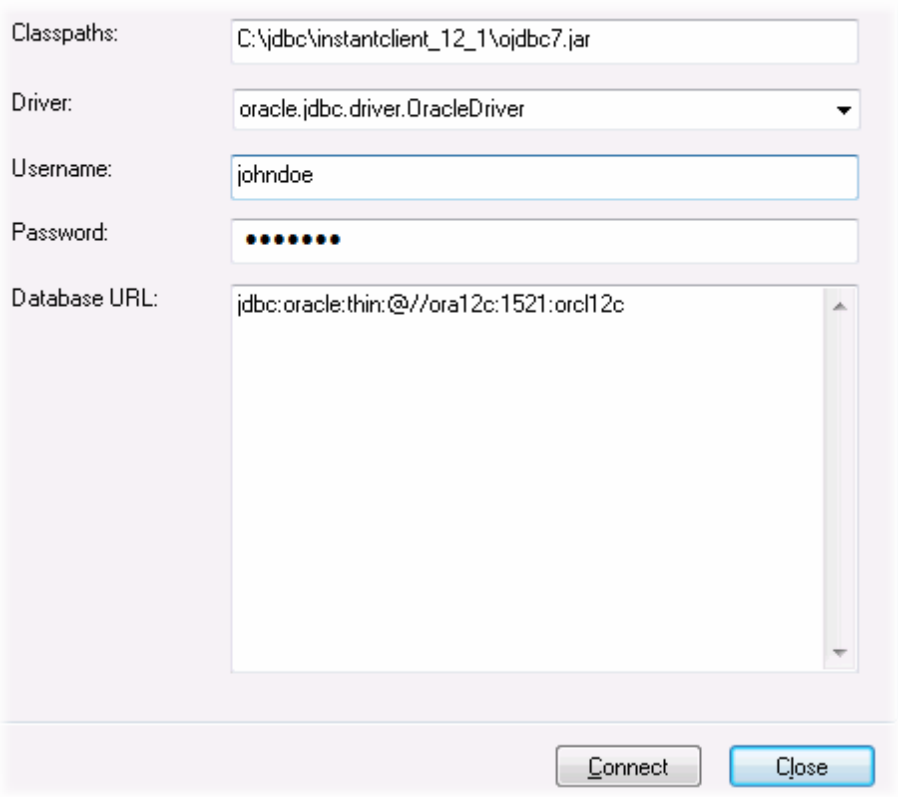
This example shows you how to connect to an Oracle database server from a client machine, using the JDBC interface. The connection is created as a pure Java connection, using the **Oracle Instant Client Package (Basic)** available from the Oracle website. The advantage of this connection type is that it requires only the Java environment and the .jar libraries supplied by the Oracle Instant Client Package, saving you the effort to install and configure a more complex database client.

Prerequisites:

- Java Runtime Environment (JRE) or Java Development Kit (JDK) must be installed on your operating system.
- 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 **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.
- You have the following database connection details: host, port, service name, username, and password.

To connect to Oracle through the Instant Client Package:

1. Start the database connection wizard.
2. Click **JDBC Connections**.
3. Next to "Classpaths", enter the path to the .jar file which provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of .jar file paths. In this example, the required .jar file is located at the following path: **C:\jdbc\instantclient_12_1\odbc7.jar**. Note that you can leave the "Classpaths" text box empty if you have added the .jar file path(s) to the CLASSPATH environment variable of the operating system (see also Configuring the CLASSPATH).
4. In the "Driver" box, select either **oracle.jdbc.OracleDriver** or **oracle.jdbc.driver.OracleDriver**. Note that these entries are available if a valid .jar file path is found either in the "Classpath" text box, or in the operating system's CLASSPATH environment variable (see the previous step).
5. Enter the username and password to the database in the corresponding text boxes.



6. Enter the connection string to the database server in the Database URL text box, by replacing the highlighted values with the ones applicable to your database server.

```
jdbc:oracle:thin:@//host:port:service
```

7. Click **Connect**.

Connecting to PostgreSQL (ODBC)

This topic provides sample instructions for connecting to a PostgreSQL database server from a Windows machine through the ODBC driver. The PostgreSQL ODBC driver is not available on Windows, so it must be downloaded and installed separately. This example uses the psqlODBC driver (version 09_03_300-1) downloaded from the official website (see also Database Drivers Overview).

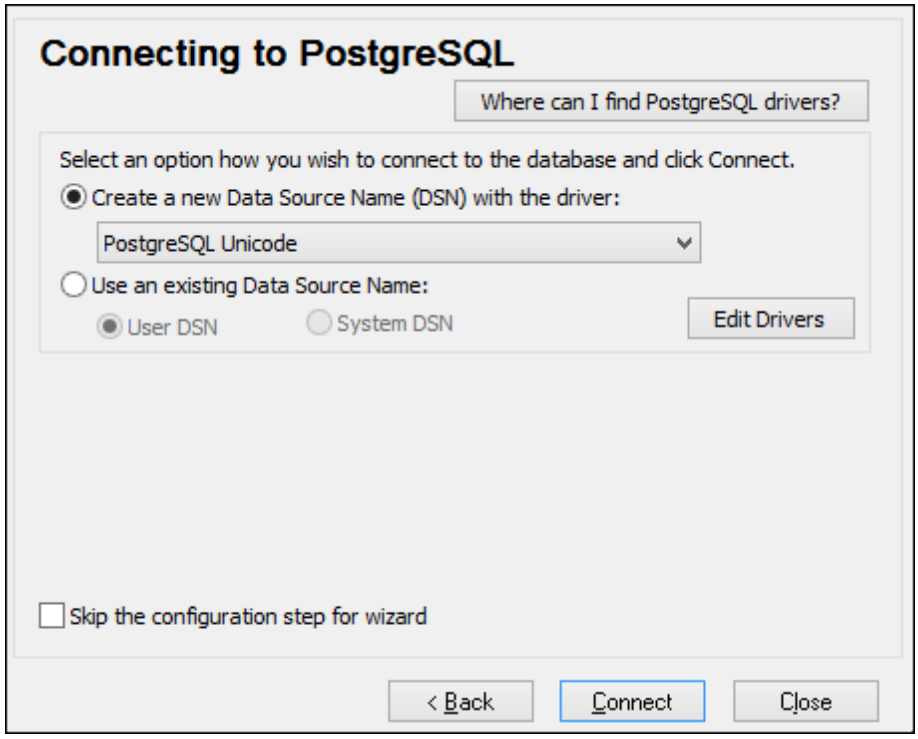
Note: You can also connect to a PostgreSQL database server directly (without the ODBC driver), see Setting up a PostgreSQL Connection.

Prerequisites:

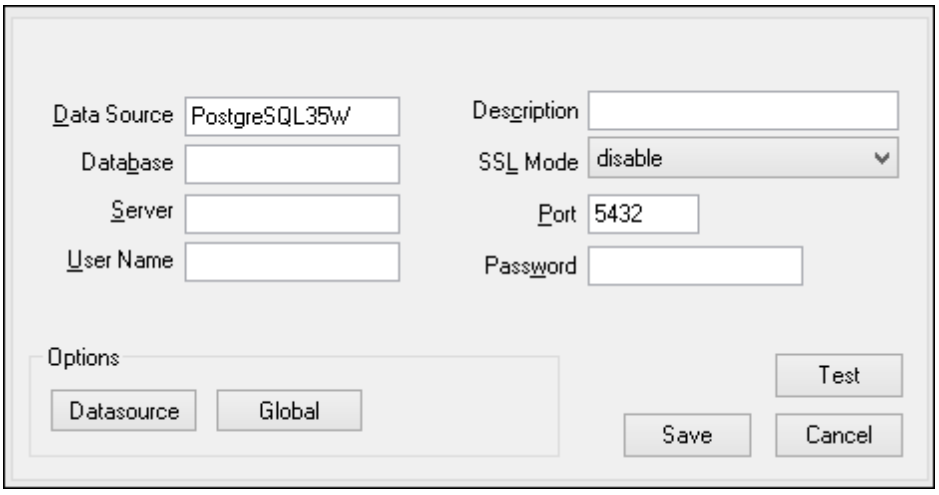
- *psqlODBC* driver must be installed on your operating system (for installation instructions, check the documentation supplied with the driver).
- You have the following database connection details: server, port, database, user name, and password.

To connect to PostgreSQL using ODBC:

1. Start the database connection wizard.
2. Select **PostgreSQL (ODBC)**, and then click **Next**.



3. Select **Create a new Data Source Name (DSN) with the driver**, and select the PostgreSQL driver. If no PostgreSQL driver is available in the list, click **Edit Drivers**, and select any available PostgreSQL drivers (the list contains all ODBC drivers installed on your operating system).
4. Click **Connect**.



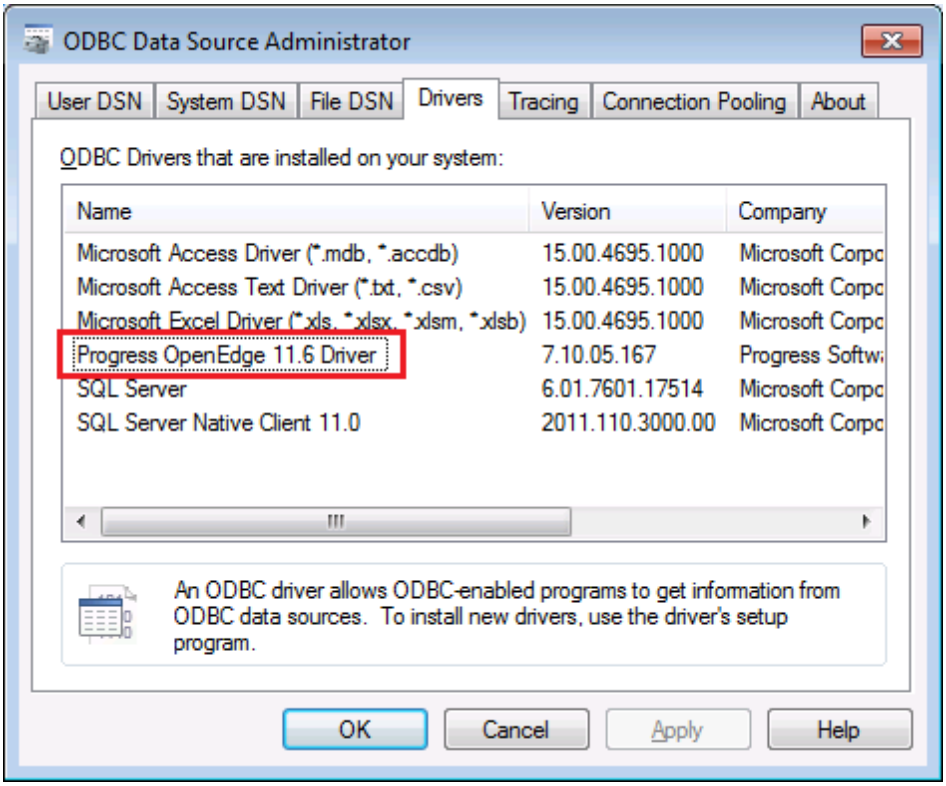
5. Fill in the database connection credentials (Database, Server, Port, User Name, Password), and then click **OK**.

Connecting to Progress OpenEdge (ODBC)

This topic provides sample instructions for connecting to a Progress OpenEdge database server through the Progress OpenEdge 11.6 ODBC driver.

Prerequisites

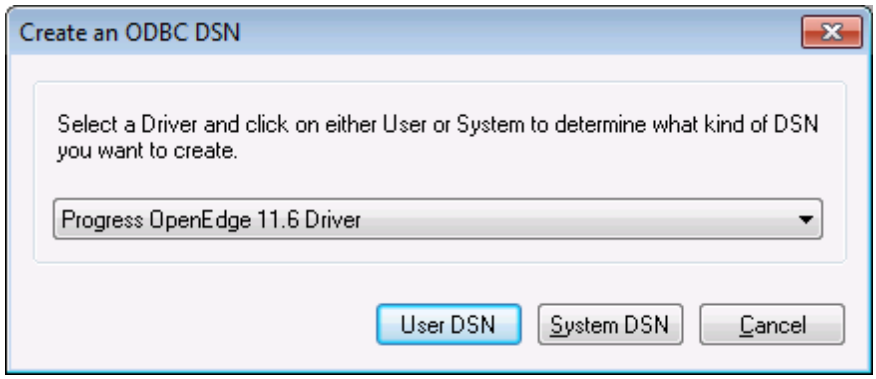
- The *ODBC Connector for Progress OpenEdge* driver must be installed on your operating system. The Progress OpenEdge ODBC driver can be downloaded from the vendor's website (see also Database Drivers Overview). Make sure to download the 32-bit driver when running the 32-bit version of MobileTogether Designer, and the 64-bit driver when running the 64-bit version. After installation, check if the ODBC driver is available on your machine (see also Viewing the Available ODBC Drivers).



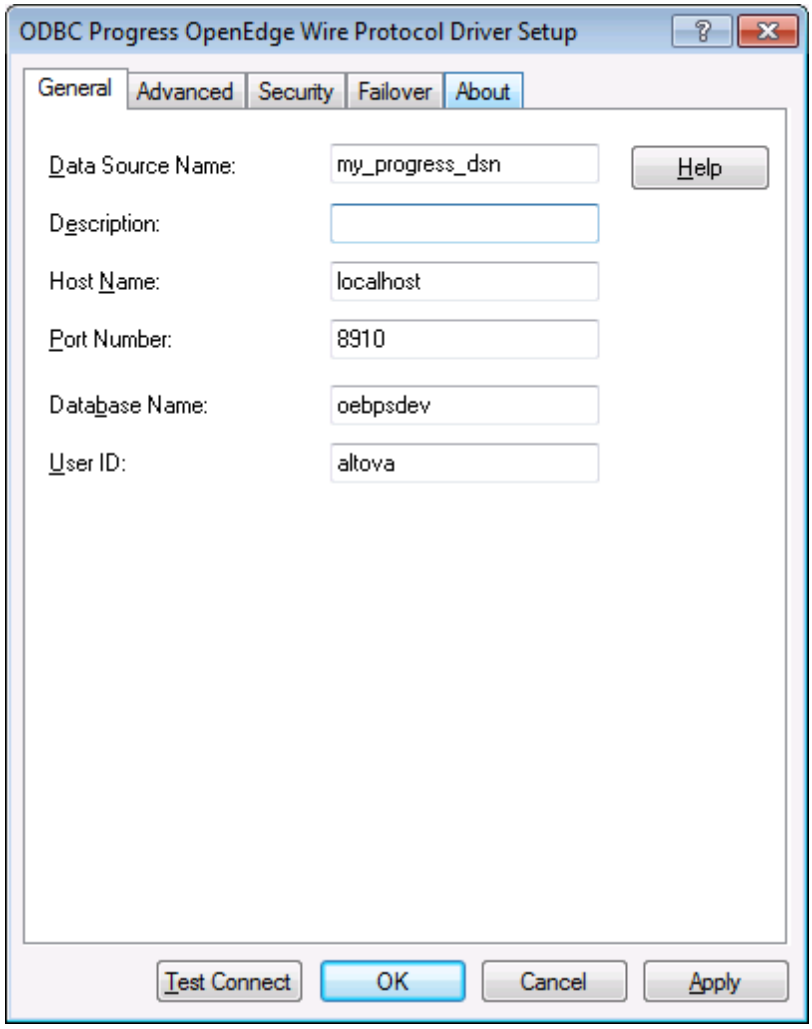
- You have the following database connection details: host name, port number, database name, user ID, and password.

Connecting to Progress OpenEdge through ODBC

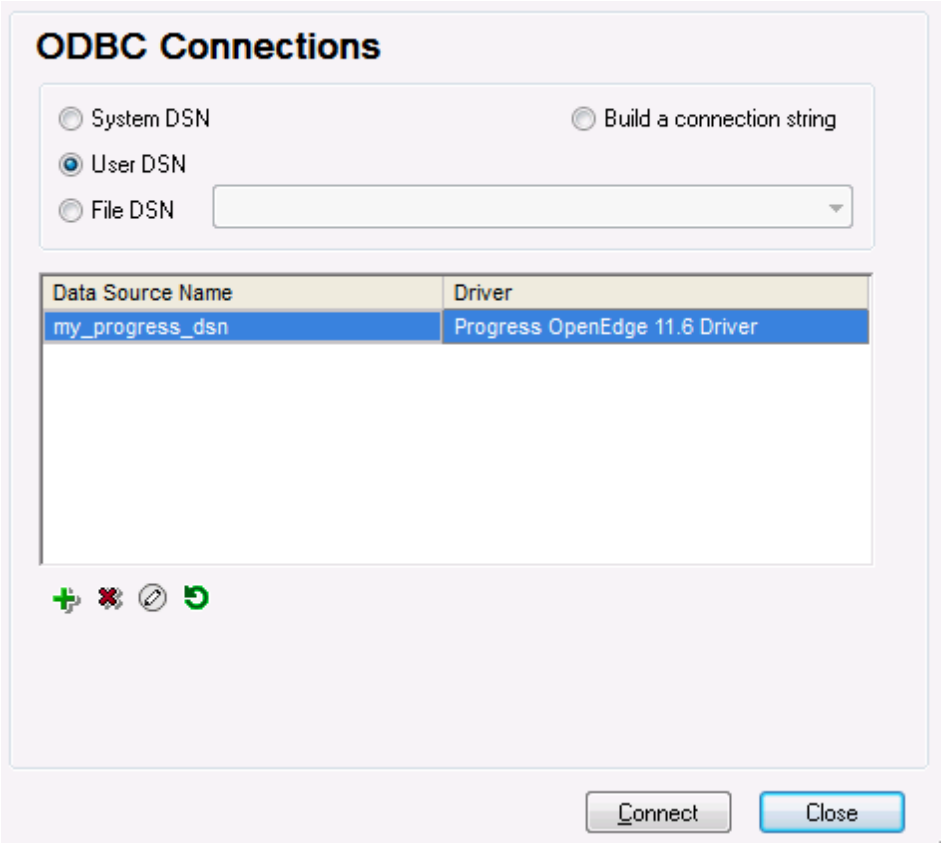
1. Start the database connection wizard.
2. Click **ODBC Connections**.
3. Click **User DSN** (alternatively, click **System DSN**, or **File DSN**, in which case the subsequent instructions will be similar).
4. Click **Add** .
5. Select the **Progress OpenEdge Driver** from the list, and click **User DSN** (or **System DSN**, if applicable).



6. Fill in the database connection credentials (Database, Server, Port, User Name, Password), and then click **OK**. To verify connectivity before saving the entered data, click **Test Connect**.



7. Click OK. The new data source now appears in the list of ODBC data sources.



8. Click **Connect**.

Connecting to Progress OpenEdge (JDBC)

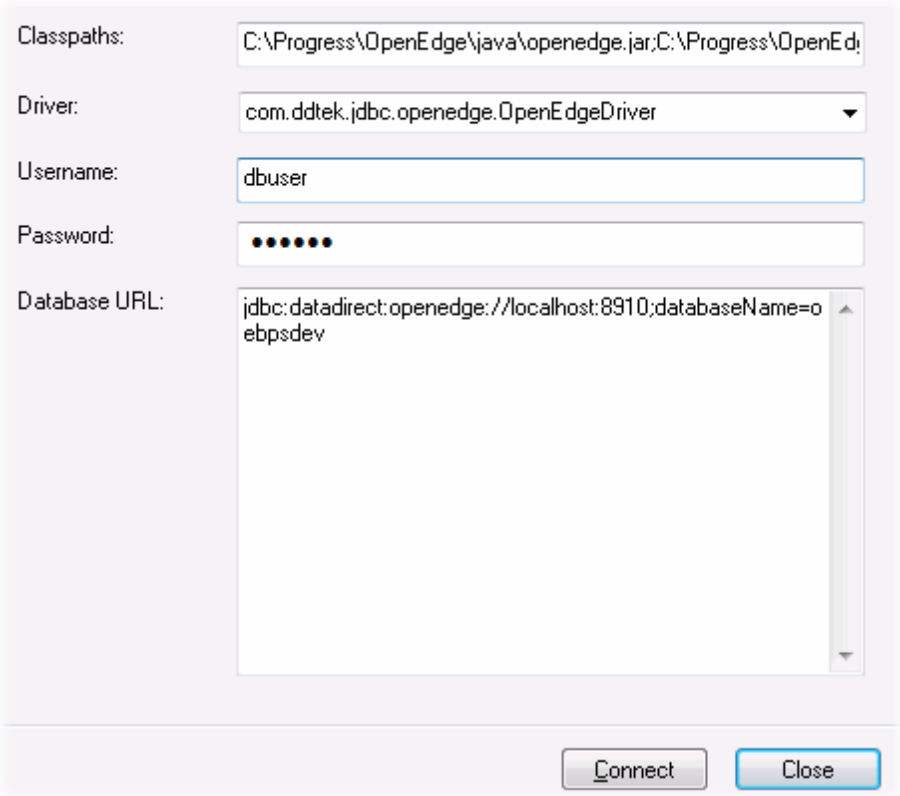
This topic provides sample instructions for connecting to a Progress OpenEdge 11.6 database server through JDBC.

Prerequisites

- Java Runtime Environment (JRE) or Java Development Kit (JDK) must be installed on your operating system. 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 name, username, and password.

Connecting to OpenEdge through JDBC

1. Start the database connection wizard.
2. Click **JDBC Connections**.
3. Next to "Classpaths", enter the path to the .jar file which provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of .jar file paths. In this example, the required .jar file paths are: **`C:\Progress\OpenEdge\java\openedge.jar;C:\Progress\OpenEdge\java\pool.jar;`**. Note that you can leave the "Classpaths" text box empty if you have added the .jar file path(s) to the CLASSPATH environment variable of the operating system (see also Configuring the CLASSPATH).
4. In the "Driver" box, select **com.ddtek.jdbc.openedge.OpenEdgeDriver**. Note that this entry is available if a valid .jar file path is found either in the "Classpath" text box, or in the operating system's CLASSPATH environment variable (see the previous step).



5. Enter the username and password to the database in the corresponding text boxes.
6. Enter the connection string to the database server in the Database URL text box, by replacing the highlighted values with the ones applicable to your database server.

```
jdbc:datadirect:openedge://host:port;databaseName=db_name
```

7. Click **Connect**.

Connecting to Sybase (JDBC)

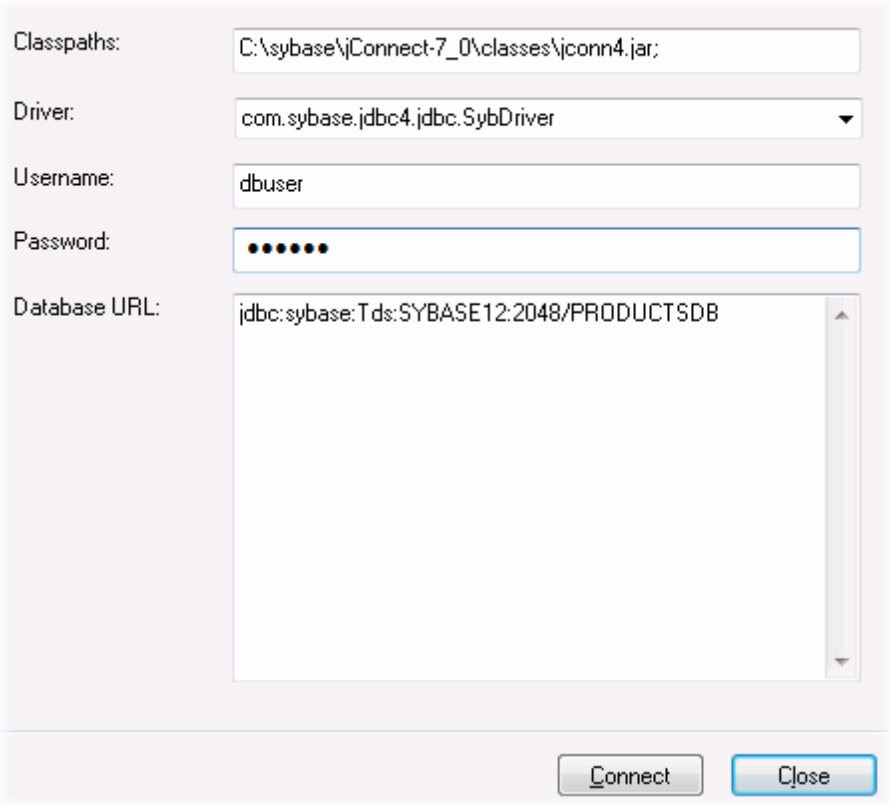
This topic provides sample instructions for connecting to a Sybase database server through JDBC.

Prerequisites:

- Java Runtime Environment (JRE) must be installed on your operating system.
- 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). For the installation instructions of the database client, refer to Sybase documentation.
- You have the following database connection details: host, port, database name, username, and password.

To connect to Sybase through JDBC:

1. Start the database connection wizard.
2. Click **JDBC Connections**.
3. Next to "Classpaths", enter the path to the .jar file which provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of .jar file paths. In this example, the required .jar file path is: **C:\sybase\jConnect-7_0\classes\jconn4.jar**. Note that you can leave the "Classpaths" text box empty if you have added the .jar file path(s) to the CLASSPATH environment variable of the operating system (see also Configuring the CLASSPATH).
4. In the "Driver" box, select **com.sybase.jdbc4.jdbc.SybDriver**. Note that this entry is available if a valid .jar file path is found either in the "Classpath" text box, or in the operating system's CLASSPATH environment variable (see the previous step).



5. Enter the username and password to the database in the corresponding text boxes.
6. Enter the connection string to the database server in the Database URL text box, by replacing the highlighted values with the ones applicable to your database server.

```
jdbc:sybase:Tds:hostName:port/databaseName
```

7. Click **Connect**.

Connecting to Teradata (ODBC)

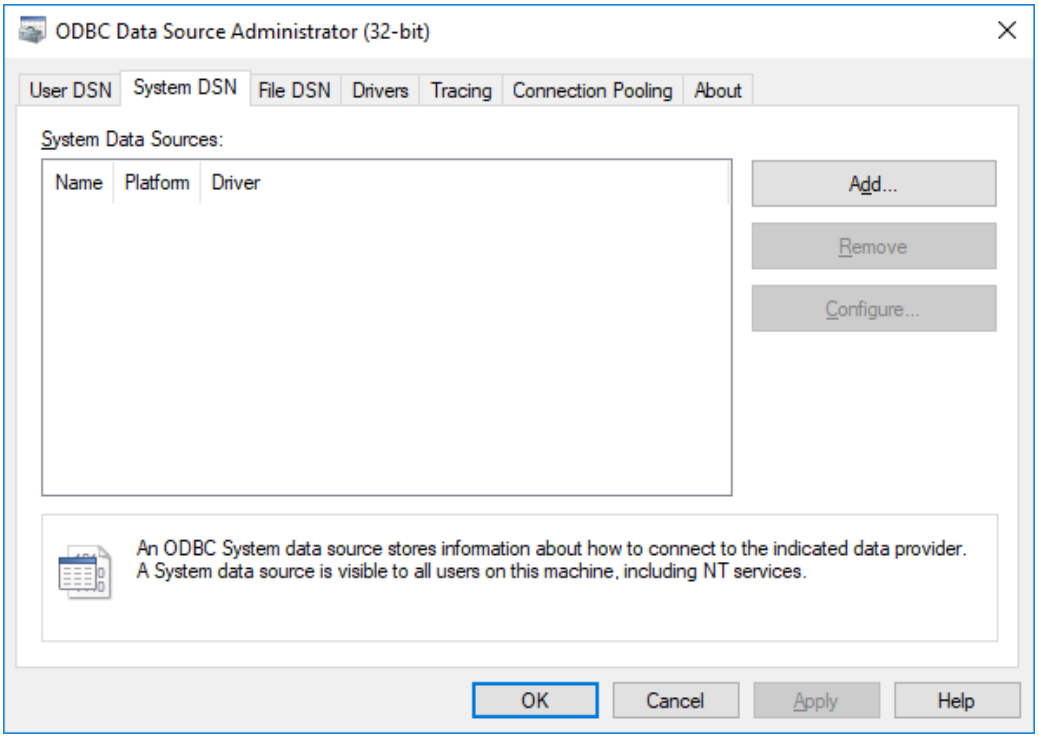
This example illustrates how to connect to a Teradata database server through ODBC.

Prerequisites:

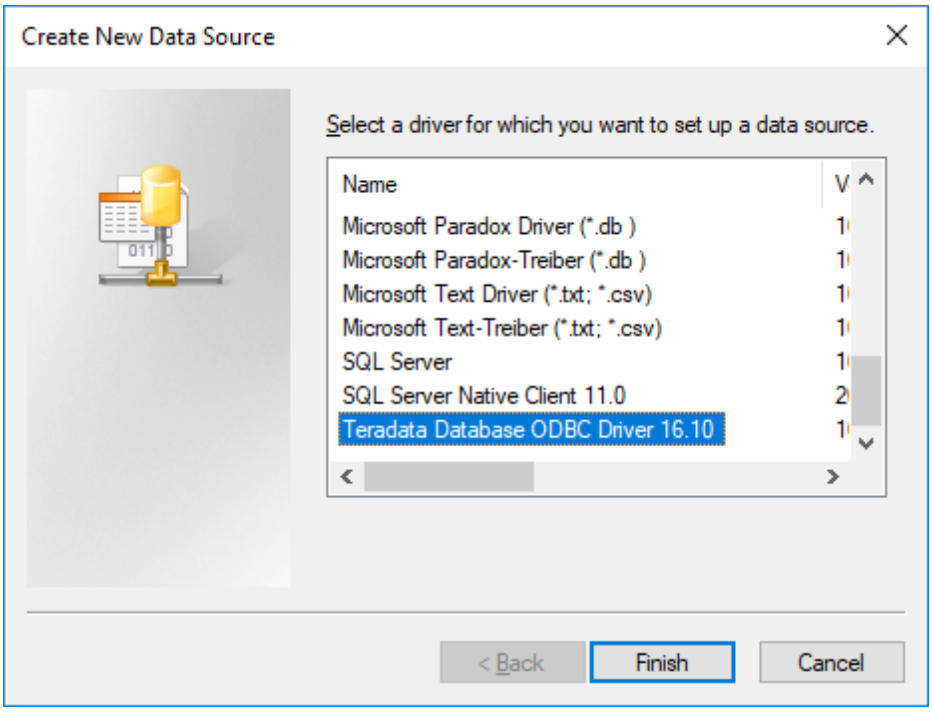
- The Teradata ODBC driver must be installed (see http://downloads.teradata.com/download/connectivity/odbc-driver/windows. This example uses Teradata ODBC Driver for Windows version 16.20.00.
- You have the following database connection details: host, username, and password.

To connect to Teradata through ODBC:

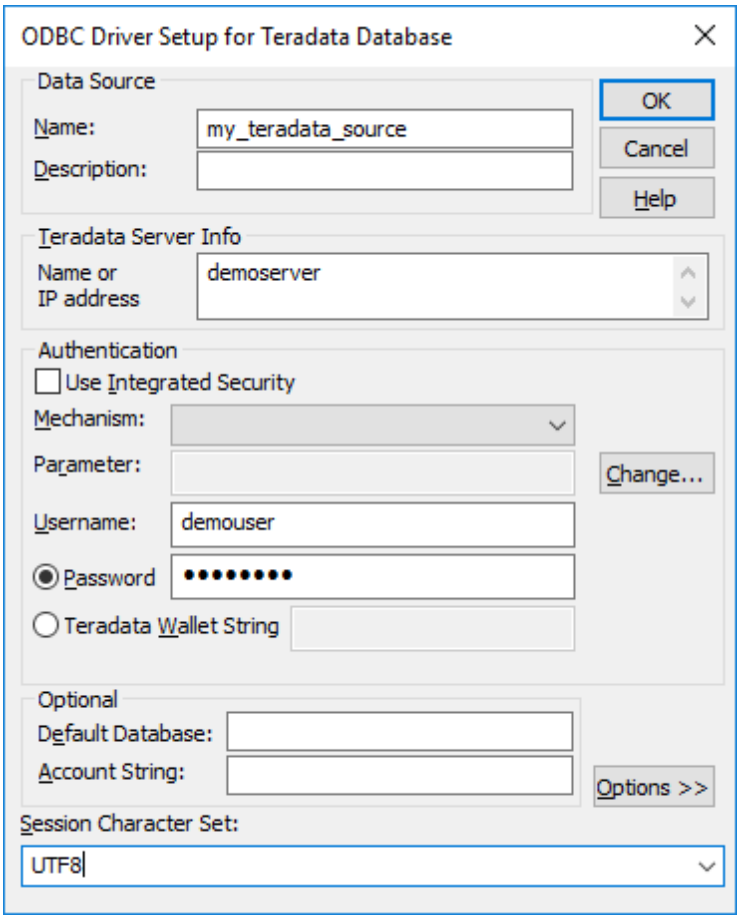
1. Press the **Windows** key, start typing "ODBC", and select **Set up ODBC data sources (32-bit)** from the list of suggestions. If you have a 64-bit ODBC driver, select **Set up ODBC data sources (64-bit)** and use 64-bit MobileTogether Designer in the subsequent steps.



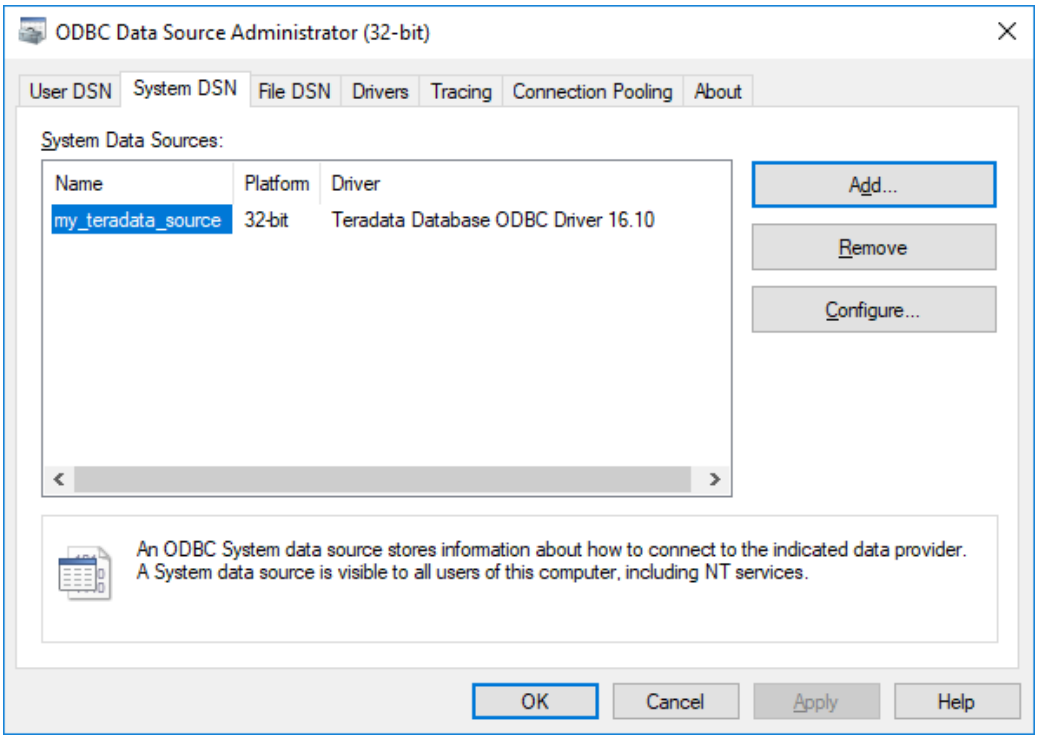
2. Click the **System DSN** tab, and then click **Add**.



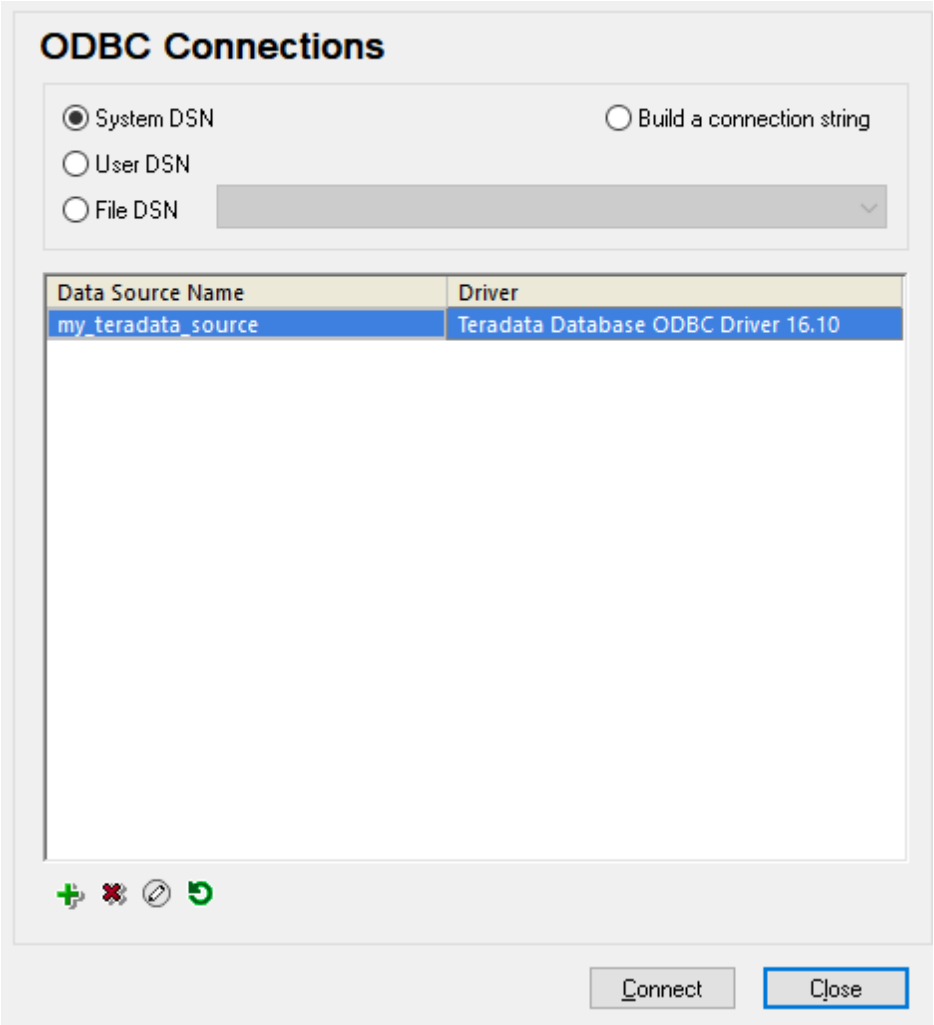
3. Select **Teradata Database ODBC Driver** and click **Finish**.



4. Enter name and, optionally, a description that will help you identify this ODBC data source in future. Also, enter the database connection credentials (Database server, User, Password), and, optionally, select a database.
5. Click **OK**. The data source now appears in the list.



6. Run MobileTogether Designer and start the database connection wizard.
7. Click **ODBC Connections**.



8. Click **System DSN**, select the data source created previously, and then click **Connect**.

Note: If you get the following error: "The driver returned invalid (or failed to return) SQL_DRIVER_ODBC_VER: 03.80", make sure that the path to the ODBC client (for example, **C:\Program Files\Teradata\Client\16.10\bin**, if you installed it to this location) exists in your system's PATH environment variable. If this path is missing, add it manually.

Connecting to Teradata (JDBC)

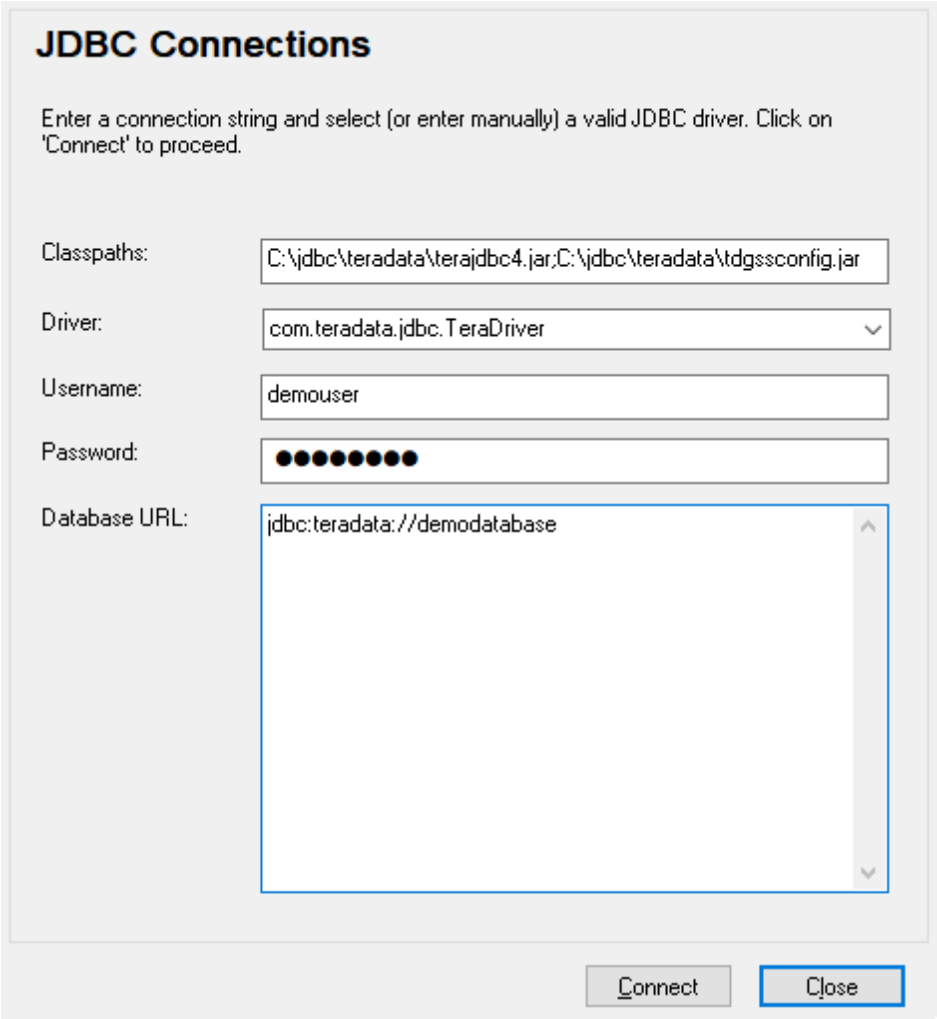
This example illustrates how to connect to a Teradata database server through JDBC.

Prerequisites:

- Java Runtime Environment (JRE) or Java Development Kit (JDK) must be installed on your operating system.
- 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 http://downloads.teradata.com/download/connectivity/jdbc-driver.
- You have the following database connection details: host, database, port, username, and password.

To connect to Teradata through JDBC:

1. Start the database connection wizard.
2. Click **JDBC Connections**.
3. Next to "Classpaths", enter the path to the .jar file which provides connectivity to the database. If necessary, you can also enter a semicolon-separated list of .jar file paths. In this example, the .jar files are located at the following path: **C:\jdbc\teradata**. Note that you can leave the "Classpaths" text box empty if you have added the .jar file path(s) to the CLASSPATH environment variable of the operating system (see also Configuring the CLASSPATH).
4. In the "Driver" box, select **com.teradata.jdbc.TeraDriver**. Note that this entry is available if a valid .jar file path is found either in the "Classpath" text box, or in the operating system's CLASSPATH environment variable (see the previous step).



5. Enter the username and password to the database in the corresponding text boxes.
6. Enter the connection string to the database server in the Database URL text box, by replacing the highlighted value with the one applicable to your database server.

```
jdbc:teradata://databaseServerName
```

7. Click **Connect**.

Database Connections on Linux and macOS

If you have licensed any of the following Altova server products—MobileTogether Server, MapForce Server, or StyleVision Server, a common scenario is to design MobileTogether designs, MapForce mappings, or StyleVision transformations on a Windows desktop machine, and then deploy them to a server machine (either Windows, Linux, or OS X / macOS) to automate their execution.

In this documentation, the term "server execution files" is used to denote the following file types:

- MapForce Server execution files (.mfx)
- MobileTogether design files (.mtd)
- StyleVision transformations (.sps) packaged as Portable XML Forms (.pxf).

The following scenarios are possible when deploying server execution files:

1. **"Design and execute on Windows"**. In this scenario, you design the MobileTogether designs, MapForce mappings, or StyleVision transformations on Windows, and then run their corresponding server execution files on a Windows system as well (which can either be the same Windows machine, or a remote Windows server).
2. **"Design on Windows, execute on Linux or macOS"**. In this scenario, you design all of the above files on Windows, and then deploy their corresponding server execution files to Linux or OS X / macOS for execution.

In the **"Design and execute on Windows"** scenario, the selection of available database technologies and drivers comprises any of ADO, ODBC, JDBC, as well as SQLite connections (see Database Drivers Overview).

In the **"Design on Windows, execute on Linux or macOS"** scenario, ADO and ODBC connections are not supported. In this scenario, you can use direct SQLite connections (see SQLite connections) and JDBC connections (see JDBC connections).

When you deploy server execution files to a server, databases are not included in the deployed package (this also applies to file-based databases such as SQLite and Microsoft Access), so a connection to them must be set up on the deployment server as well. In other words, the same database configuration must be in place both on the operating system where you design and on the server to which you deploy the files. An exception to this rule are native (not driver-based) PostgreSQL connections. Native PostgreSQL connections do not require configuration outside MobileTogether Designer. For more information, see Setting up a PostgreSQL Connection.

In general, the scenario in which you deploy server execution files to a different operating system is slightly more complex, since it requires that the same database configuration exist on both machines. To bypass complexity while designing locally and deploying remotely, consider using the Global Resources feature available in MapForce, MobileTogether Designer, and StyleVision.

For example, you can define two different Global Resource configurations to connect to the same database: one which would specify the connection settings using the Windows-style path conventions, and another one—using Linux-style path conventions. You could then use the first connection to test your files during the design phase, and the second connection to run the execution file on the Linux server.

SQLite connections on Linux and macOS

There is no need to separately install SQLite on Linux and macOS since support for it is integrated into Altova server products as well. Therefore, if your server execution files include calls to a SQLite database, you will be able to run them without having to install SQLite first. You need to ensure, however, that the server execution files use the correct path to the database file on the Linux or OS X / macOS machine. That is, before running the server execution files on the Linux or OS X / macOS server, make sure that the SQLite database file is referenced through a path which is POSIX (Portable Operating System Interface) compliant. This assumes that no Windows-style drive letters are used in the path, and directories are delimited by the forward slash character (/). For example, the path **/usr/local/mydatabase.db** is POSIX compliant, while the path **c:\sqlite \mydatabase.db** isn't.

JDBC connections on Linux and macOS

To set up a JDBC connection on Linux or macOS:

1. Download the JDBC driver supplied by the database vendor and install it on the operating system. Make sure to select the 32-bit version if your operating system runs on 32-bit, and the 64-bit version if your operating system runs on 64-bit.
2. Set the environment variables to the location where the JDBC driver is installed. Typically, you will need to set the CLASSPATH variable, and possibly a few others. To find out which specific environment variables must be configured, check the documentation supplied with the JDBC driver.

Note: On macOS, the system expects any installed JDBC libraries to be in the **/Library/Java/Extensions** directory. Therefore, it is recommended that you unpack the JDBC driver to this location; otherwise, you will need to configure the system to look for the JDBC library at the path where you installed the JDBC driver.

Oracle Connections on OS X Yosemite

On OS X Yosemite, you can connect to an Oracle database through the **Oracle Database Instant Client**. Note that, if you have a Mac with a Java version prior to Java 8, you can also connect through the **JDBC Thin for All Platforms** library, in which case you may disregard the instructions in this topic.

You can download the Oracle Instant Client from the Oracle official download page. Note that there are several Instant Client packages available on the Oracle download page. Make sure to select a package with Oracle Call Interface (OCI) support, (for example, Instant Client Basic). Also, make sure to select the 32-bit version if your operating system runs on 32-bit, and the 64-bit version if your operating system runs on 64-bit.

Once you have downloaded and unpacked the Oracle Instant Client, edit the property list (.plist) file shipped with the installer so that the following environment variables point to the location of the corresponding driver paths, for example:

Variable	Sample Value
`CLASSPATH`	`/opt/oracle/instantclient_11_2/ojdbc6.jar:/opt/oracle/instantclient_11_2/ojdbc5.jar`
`TNS_ADMIN`	`/opt/oracle/NETWORK_ADMIN`
`ORACLE_HOME`	`/opt/oracle/instantclient_11_2`
`DYLD_LIBRARY_PATH`	`/opt/oracle/instantclient_11_2`
`PATH`	`$PATH:/opt/oracle/instantclient_11_2`

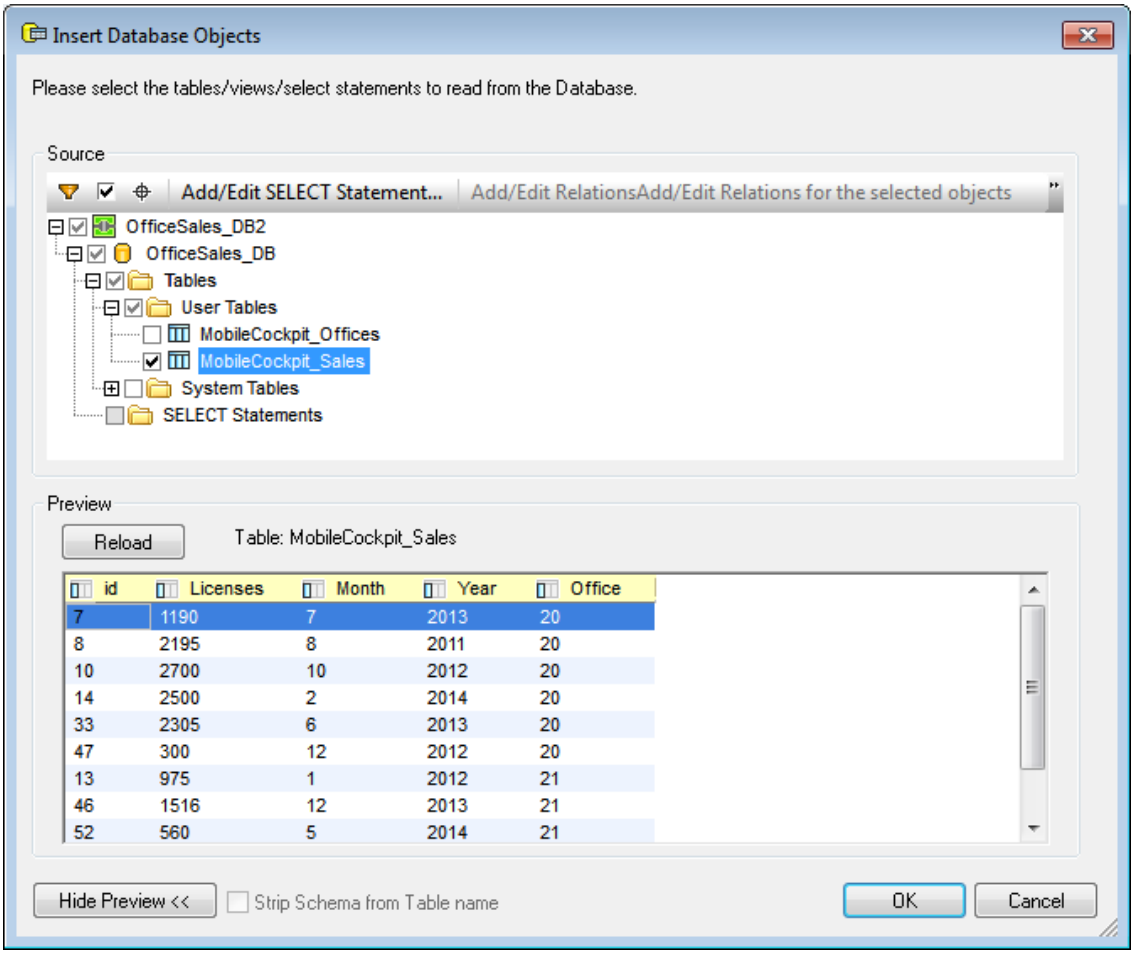
Note: Edit the sample values above to fit the paths where Oracle Instant Client files are installed on your operating system.

13.3 Selecting DB Objects as Data Sources

After connecting to a DB, the Insert Database Objects dialog (*screenshot below*) is displayed. In this dialog, you can select the DB objects you wish to add as a data source, either by selecting a table or by using a SELECT statement.

Selecting a table as a data sources

In the screenshot below, the `MobileCockpit_Sales` table has been selected. Click **Show Preview** to display the rows of the selected table in the Preview pane at the bottom of the dialog.



Note: Only one table can be selected at a time as a data source.

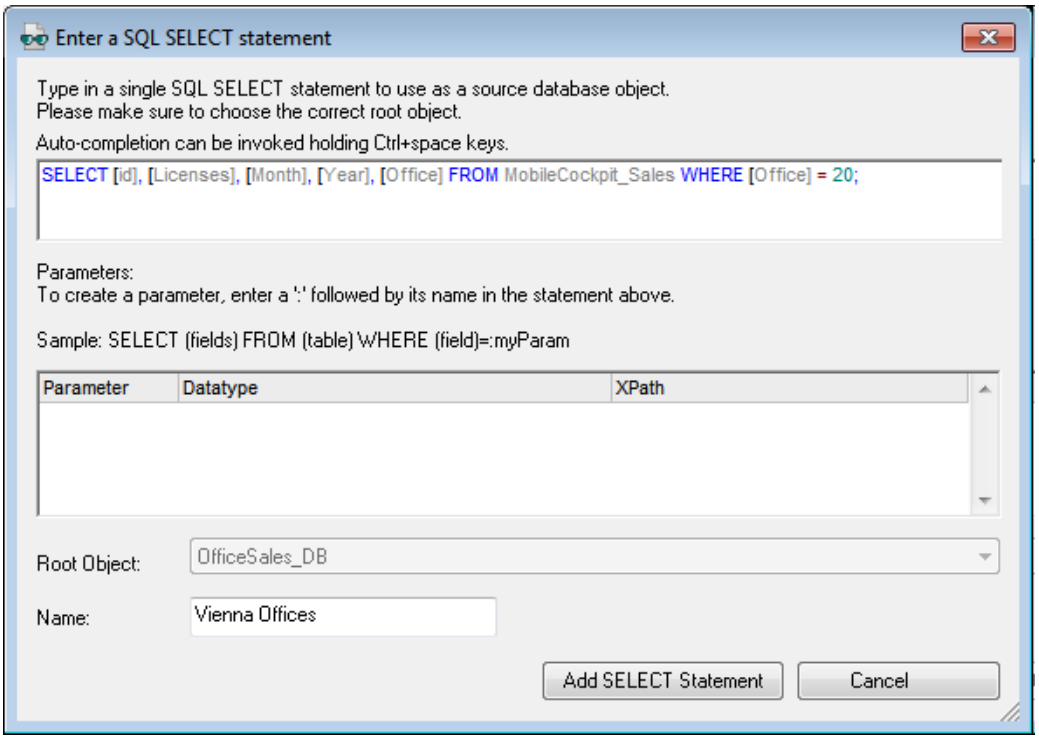
Using a SELECT statement

To select the data source by using a SELECT statement, do the following:

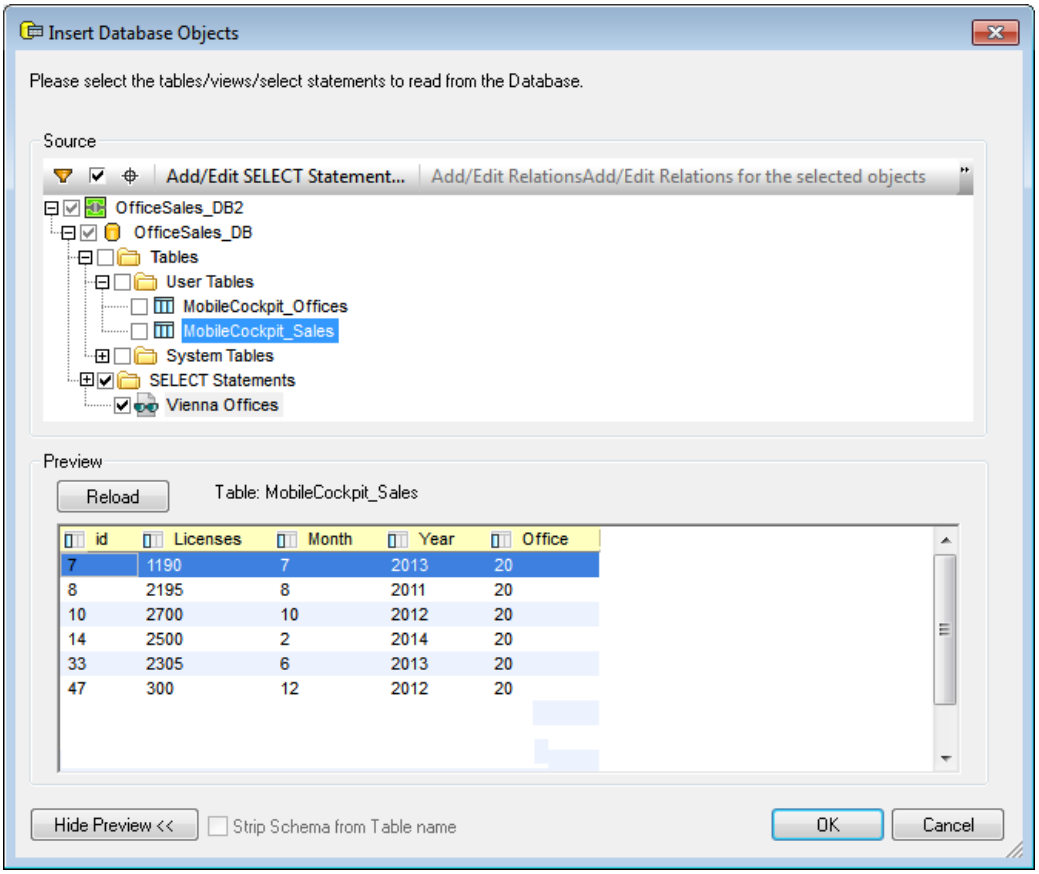
1. In the Insert Database Objects dialog (*screenshot above*), click **Add/Edit SELECT**

Statement. This displays the Enter a SQL SELECT Statement dialog (*screenshot below*).

2. Enter the `SELECT` statement you want and give the statement a name. In the screenshot below, we select, with the `WHERE` clause, rows that have `id=20`. We give the statement a name of `Vienna Offices`.



3. Click **Add SELECT Statement** to add the statement. The `SELECT` statement appears in the Source pane (*see screenshot below*).
4. Click **Show Preview** to display the rows of the selected table in the Preview pane



5. Click **OK** to add the data source to the page sources.

13.4 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 data 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.

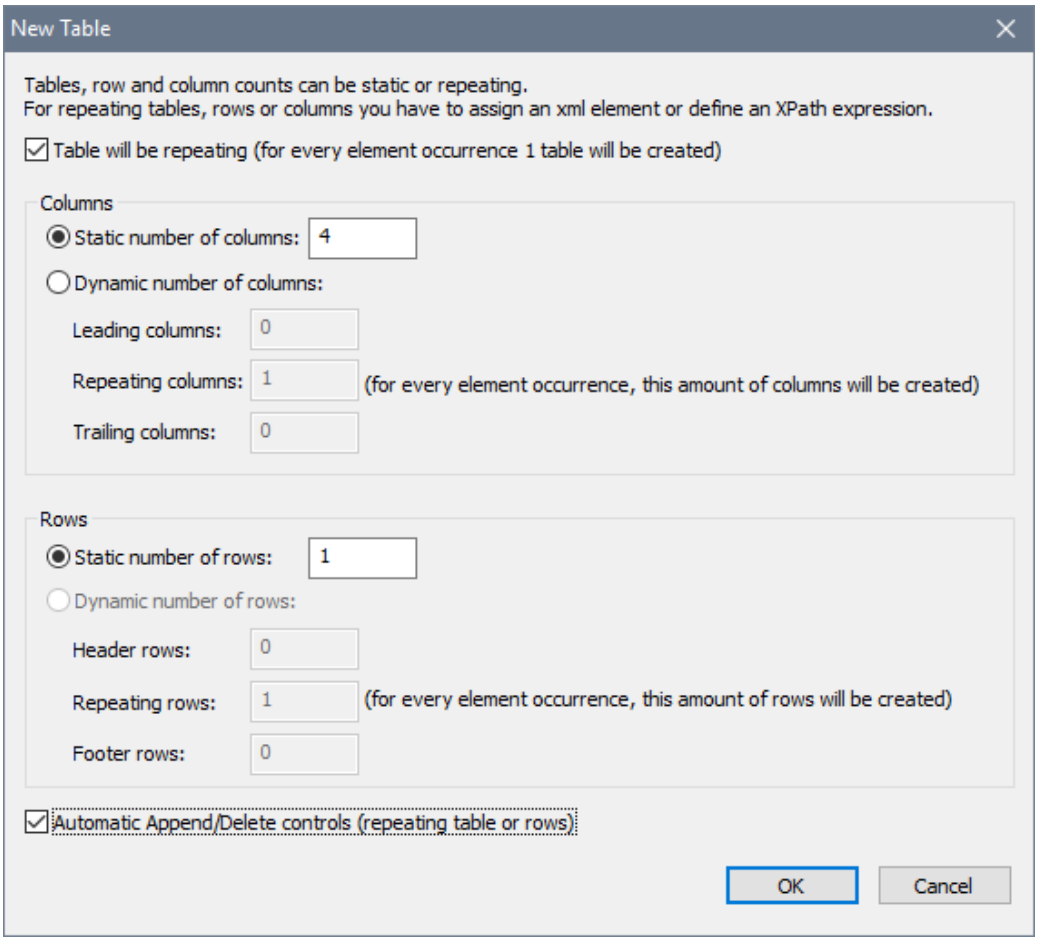
To create an `OriginalRowSet` for a page source, right-click the root node of the page source and toggle on the command **Create OriginalRowSet**.

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. User modified data is saved in the main structure created from the data source. When modified data is saved back to the DB, the `OriginalRowSet` structure is modified to contain the data newly saved to the DB. 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.

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 in the Database-And-Charts tutorial for an example of how to do this. Also see the description of how to work with tables.
- On the control, create a page source link to the data source node that is to be edited. Do this by dragging the data 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*).



The advantage of using a table with repeating rows that are linked to the repeating `Row` elements of a DB data 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. For examples, see the Database-And-Charts tutorial and the MobileTogether Designer DB examples files.

Updating, inserting, appending, and deleting nodes

The Update Data actions enable nodes in DBs to be edited when a page or control event is triggered.

- Update Node(s): Updates one or more nodes, such as DB column, with value/s generated or obtained by the action's XPath expression.
- Insert Node(s): 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): 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): 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 for the end user modifications to be passed to the DB.

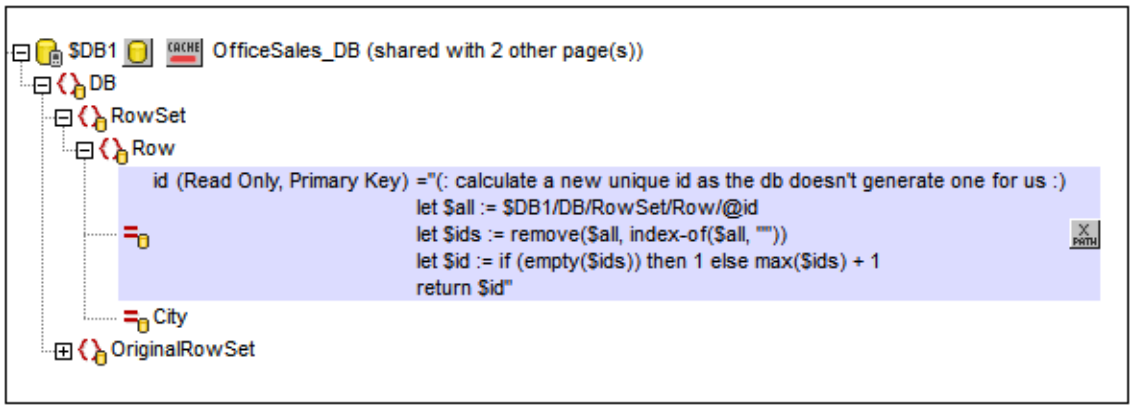
The DB Execute action and `$MT_DBExecute_Result` variable

The DB Execute action enables you to use powerful SQL statements, including `INSERT`, `APPEND`, `UPDATE`, and `DELETE`, to modify a DB. It is different from the actions listed in the previous section 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 a Save mechanism must be used to save the modifications to the DB.

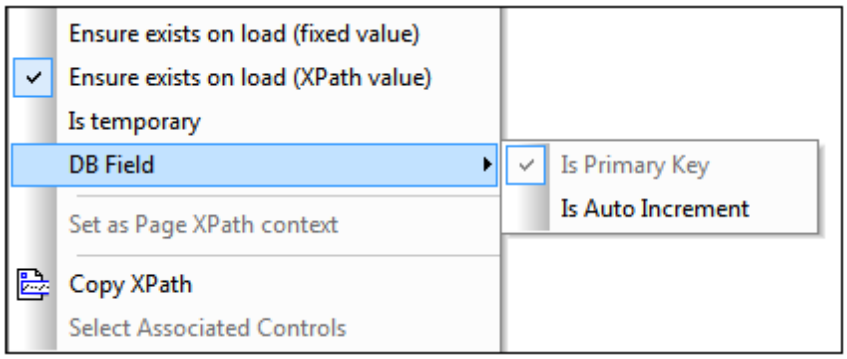
After the DB Execute action executes an SQL statement, it stores the result in a variable named **`$MT_DBExecute_Result`**. This variable can then be used in XPath expressions anywhere within the project. Consequently, structures and data from one DB can be selected (optionally based on parameters), 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*).

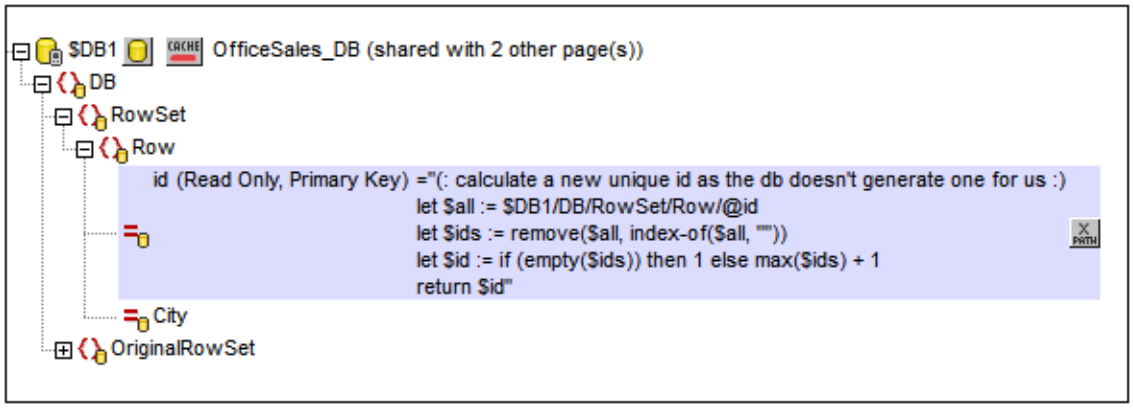


If auto-retrieval of this information was not successful, the context menu of tree nodes contains toggle commands that enable you to correctly annotate nodes (*see screenshot below*).



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
```



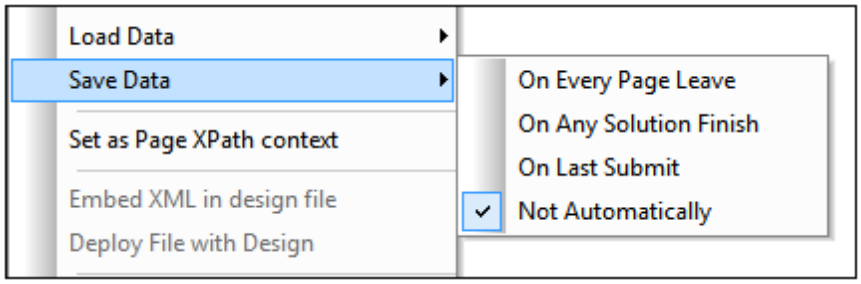
13.5 Saving Data to the DB

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 saved 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, Save to File, or Save to HTTP/FTP actions to save data.

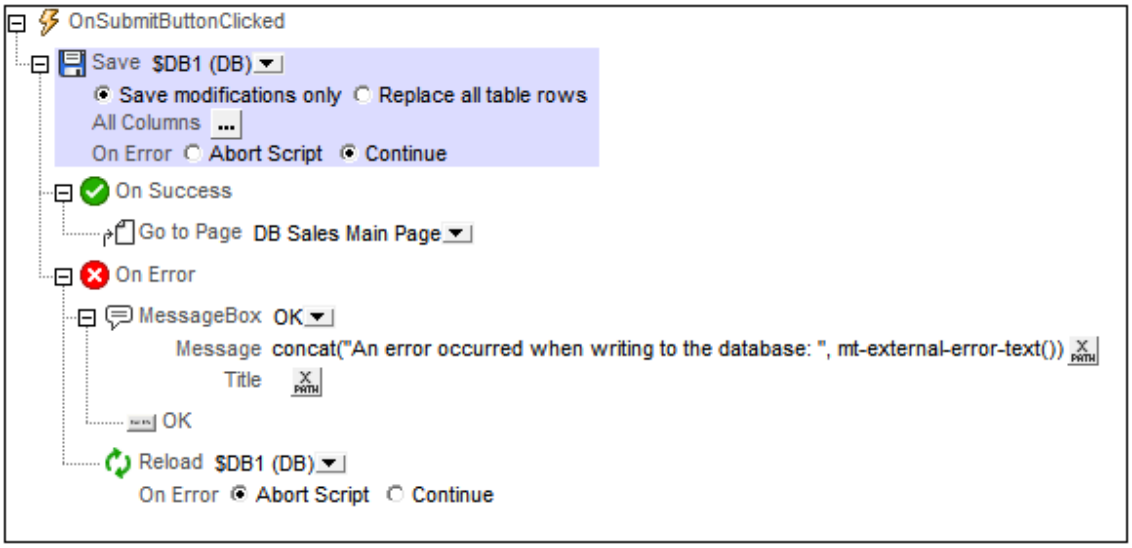
The default is *Not Automatically*.

The Save action

Data can be saved to the DB when a page event or control event is triggered for which the *Save*

action is 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.

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.



Note: If the DB has a private key, the private key is used to save only those records that have been modified, added, or deleted. If the DB has no private key, the entire modified table will be saved to the DB, replacing the original table.

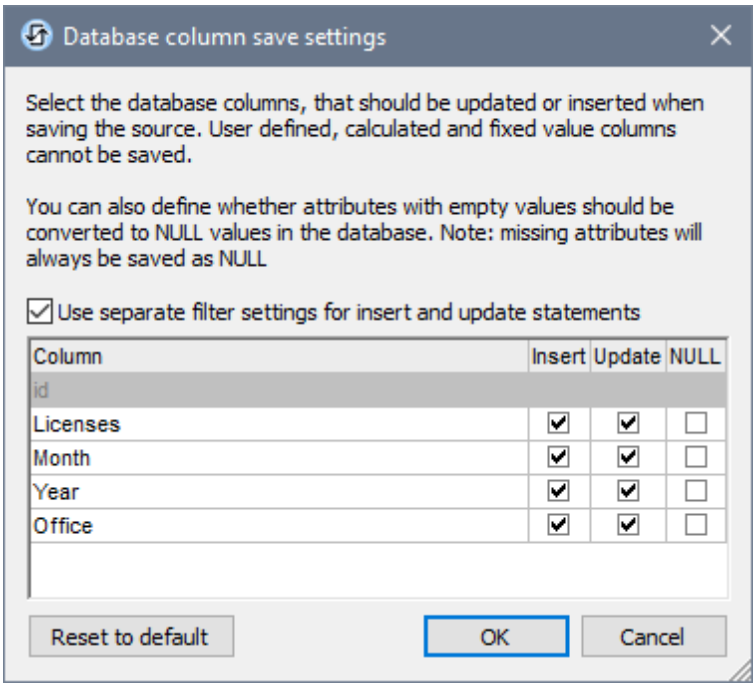
The DB Execute action

The DB Execute action 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. .

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.



The dialog displays the columns of the DB data 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. 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 data 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.

To create an `OriginalRowSet` for a page source, right-click the root node of the 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. User modified data is saved in the main structure created from the data source. When modified data is saved back to the DB, the `OriginalRowSet` structure is modified to contain the data newly saved to the DB. 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.

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 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.

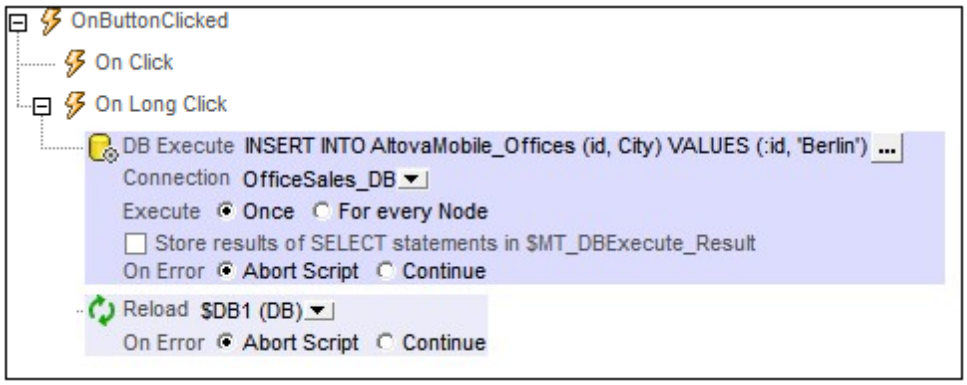
If you begin a transaction, all DB operations belonging to the same DB connection will use this transaction.

Committing a transaction makes changes visible to the world outside your transaction. Changes can be rolled back. 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.

13.6 The DB Execute Action

The DB Execute action (*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.

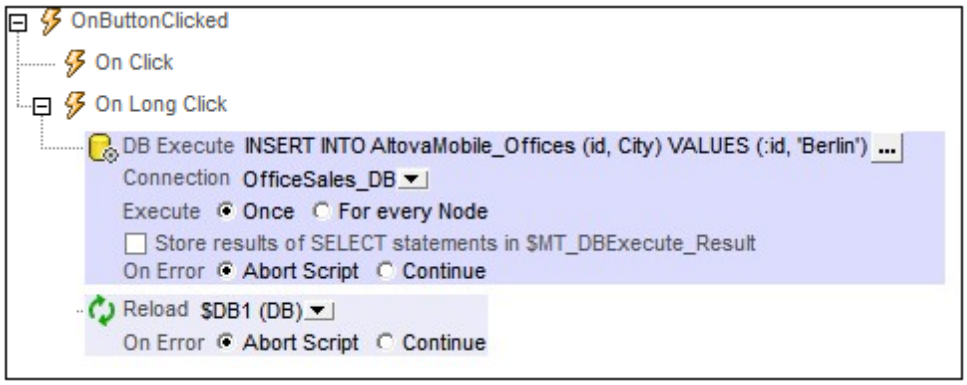


In this section, we describe how to insert, update, delete, and save data using DB Execute. The command to modify DB data is specified in the SQL statement of the action (*see screenshot above*). For a detailed description of the settings of the DB Execute action, see the section Page Design > Actions > Database > DB Execute. 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 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.



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

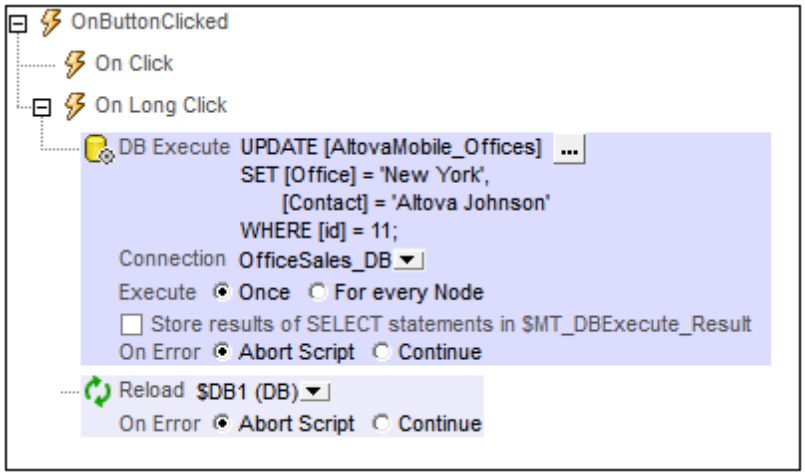
The `UPDATE` statement of SQL can be used to update rows in a database table. The `UPDATE` statement has three parts:

- 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.



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

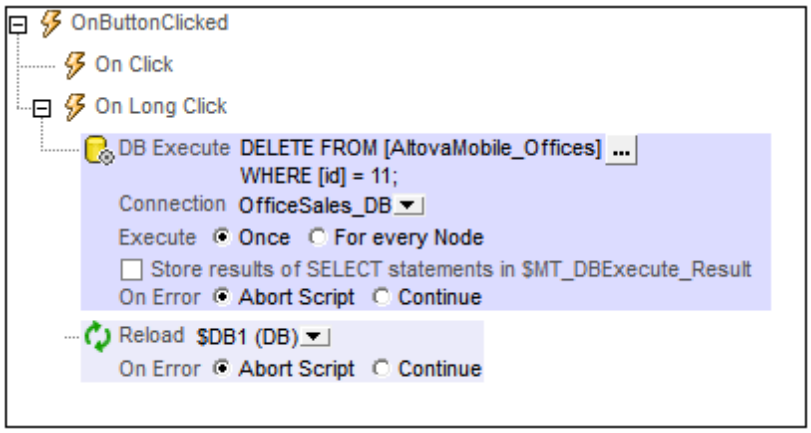
The `DELETE` statement of SQL can be used to delete:

- 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.



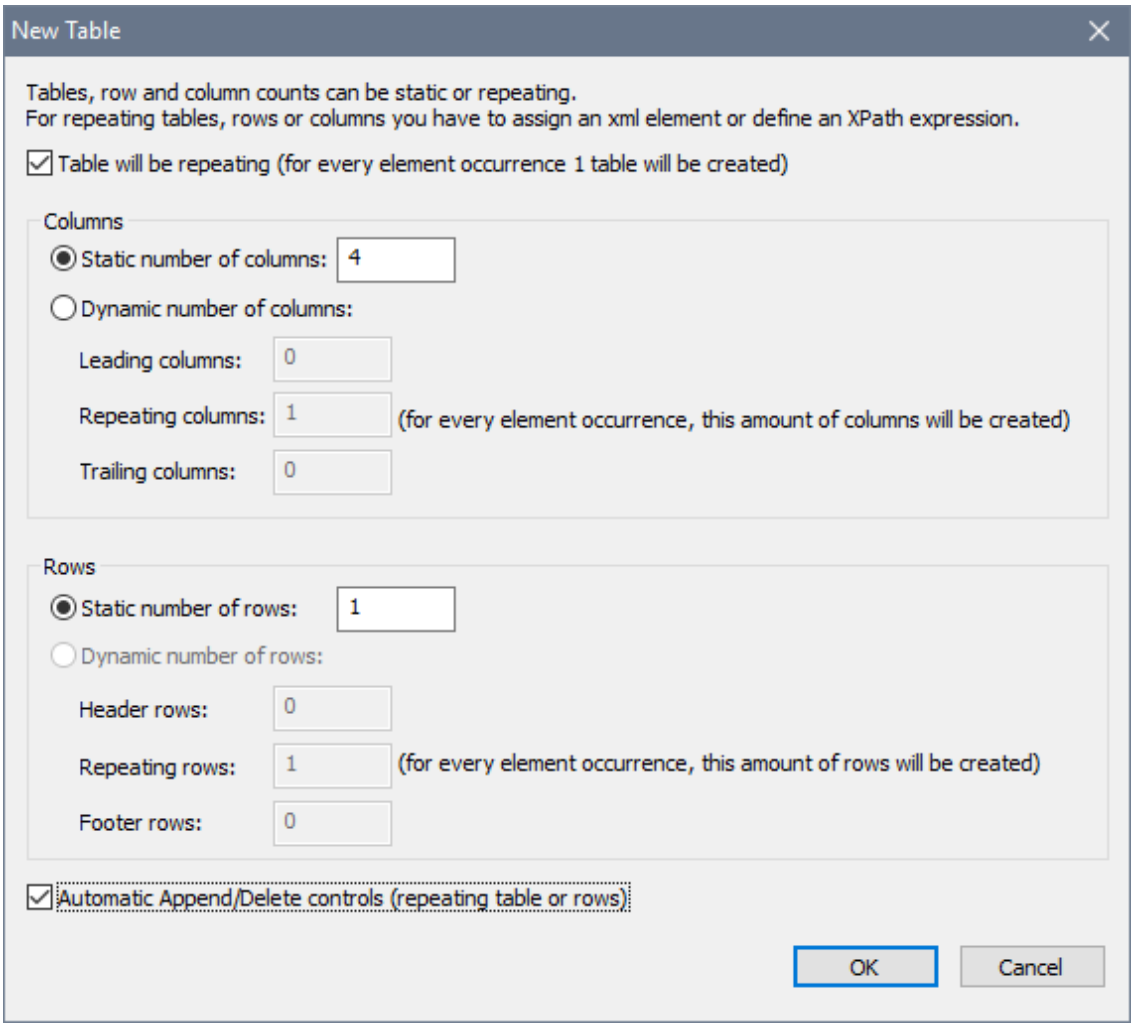
An SQL DELETE statement in a DB Execute action.

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.

13.7 Displaying DB Data

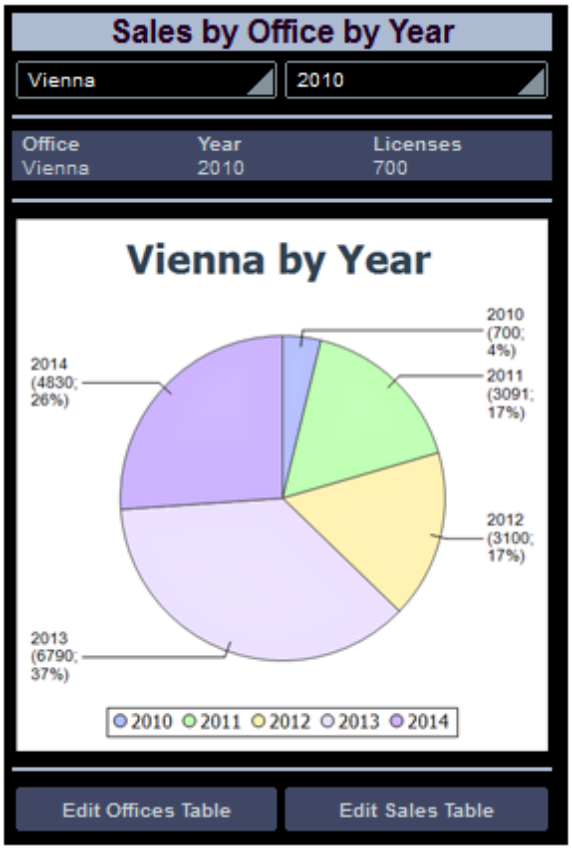
Displaying DB data in tables and other controls

DB data can be displayed in a control by creating a page source link from the control to a data 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.



Displaying DB data as charts

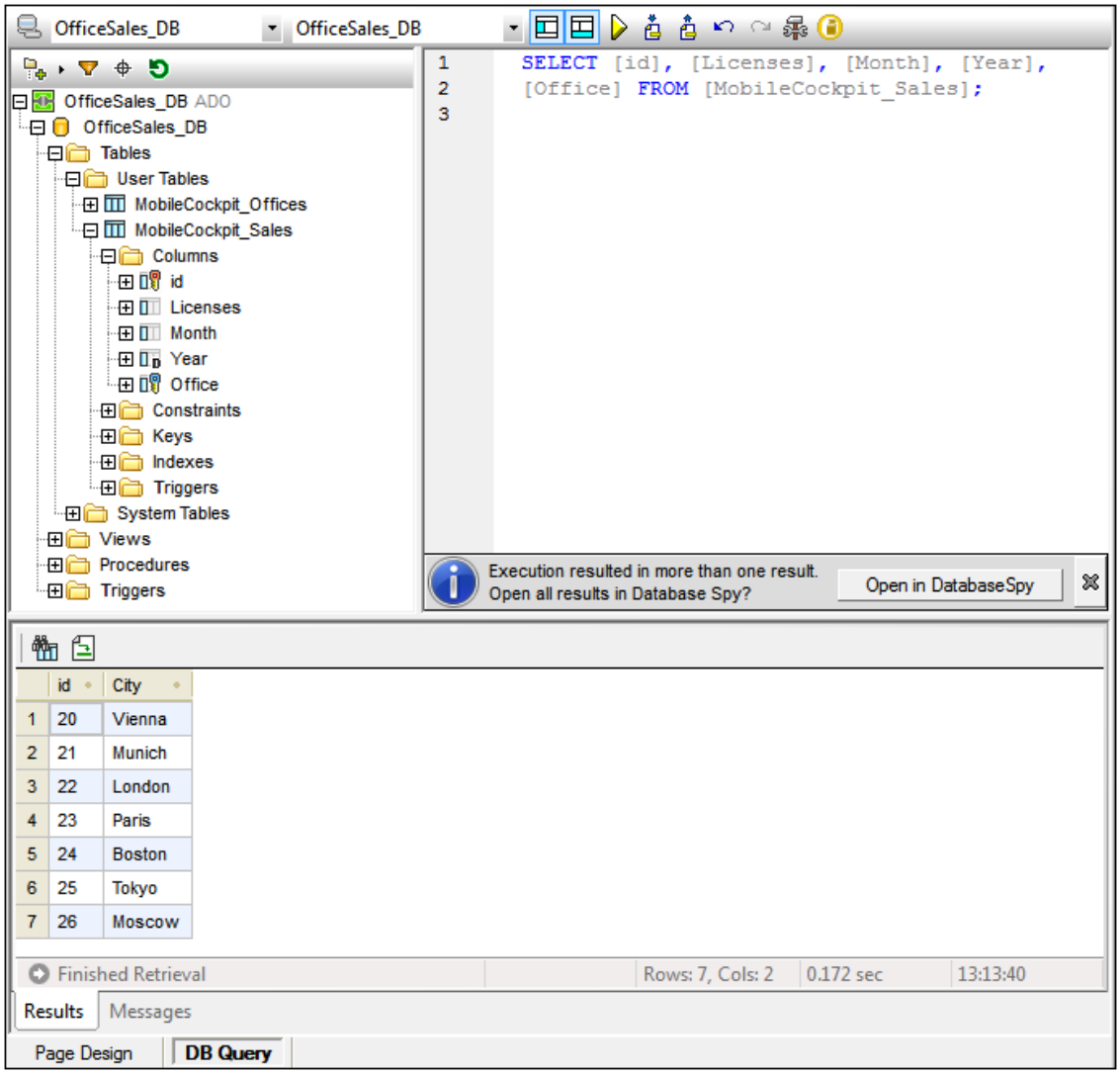
In addition to being able to display DB data directly, you can also create charts based on DB data.



For an example of how to do this, see the tutorial, Database-And-Charts.

13.8 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.



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 via the *Database Query* | *Connect to a Data Source* window.
2. The connected database or parts of it are displayed in the Browser pane, which can be configured to suit viewing requirements.
3. A query written in a syntax appropriate to the database to be queried is entered in the Query pane, and the query is executed.
4. The results of the query can be viewed through various filters.

Supported databases

The following databases are supported. The available root object for each database is also listed. While Altova endeavors to support other databases, successful connection and data processing have only been tested with the databases listed below. 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	Root Object	Notes
Firebird 2.5.4	database	
IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5	schema	
IBM DB2 for i 6.1, 7.1	schema	Logical files are supported and shown as views.
IBM Informix 11.70	database	
MariaDB 10	database	
Microsoft Access 2003, 2007, 2010, 2013	database	
Microsoft Azure SQL Database	database	SQL Server 2016 codebase
Microsoft SQL Server 2005, 2008, 2012, 2014, 2016	database	
MySQL 5.0, 5.1, 5.5, 5.6, 5.7	database	
Oracle 9i, 10g, 11g, 12c	schema	
PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0.10, 9.1.6, 9.2.1, 9.4, 9.5, 9.6	database	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	database	
SQLite 3.x	database	SQLite connections are supported as native, direct connections to the SQLite database file.

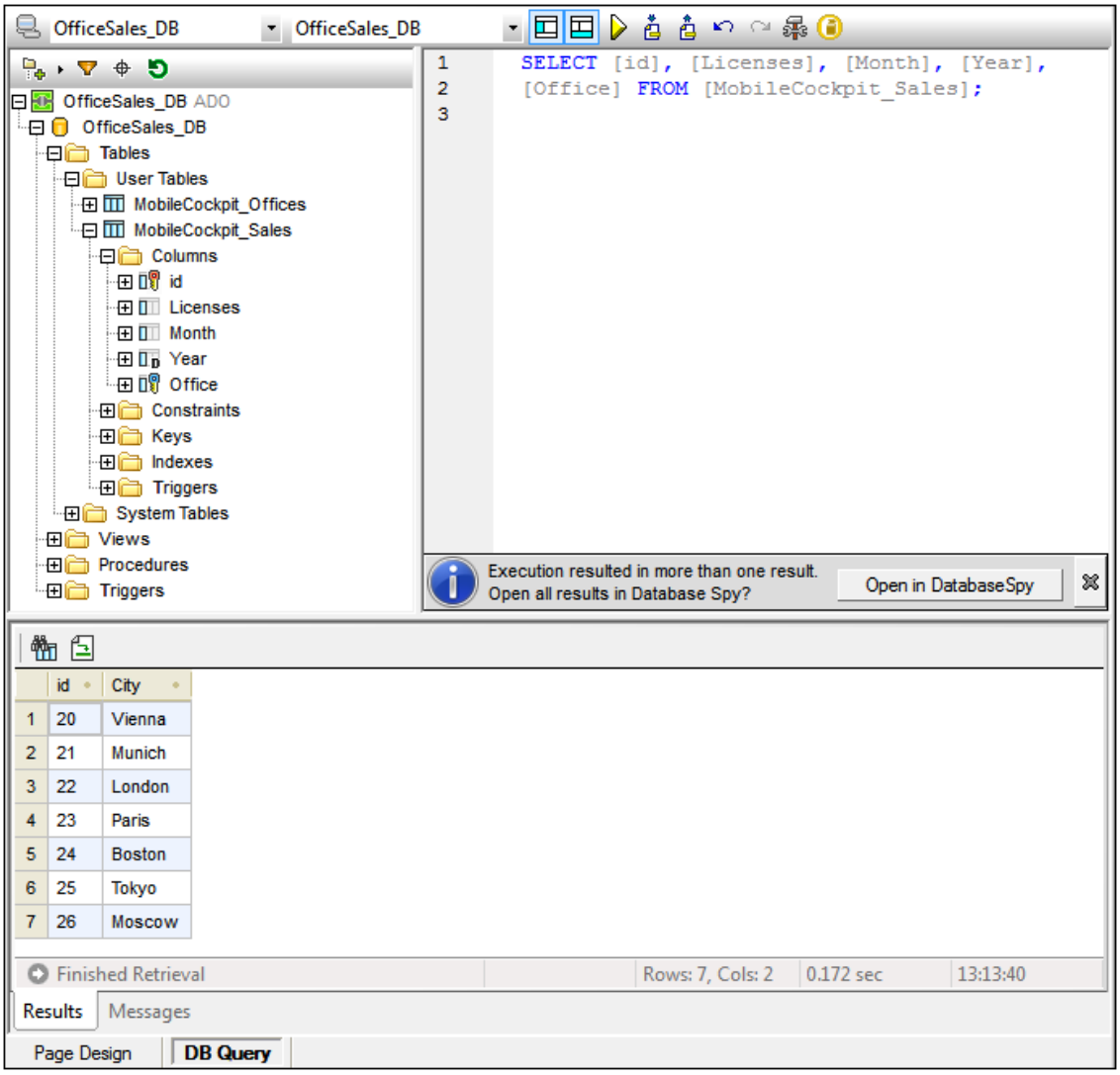
Database	Root Object	Notes
		No separate drivers are required.
Sybase ASE 15, 16	database	
Teradata 16	database	

GUI Overview and Toolbar

The **DB Query View** (*screenshot below*) is divided into the following parts:

- The Browser window at left, which displays connection info and database tables
- The SQL Editor window (Query window), to the right of the Browser window, contains your SQL queries
- The Results tab of the Result/Messages windows displays the query results in tabular form
- The Messages tab of the Result/Messages windows displays warnings or error messages

These windows are described in detail in the sub-sections of this section.



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.

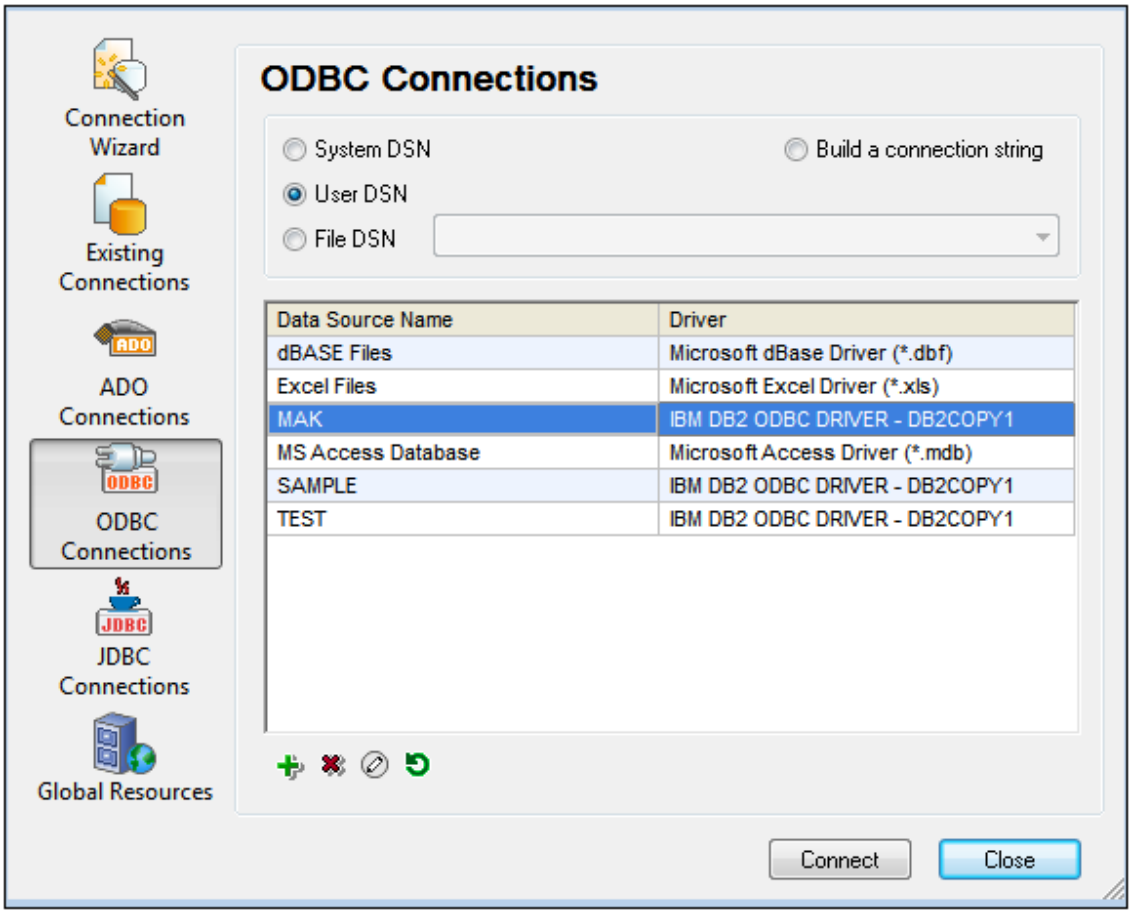
Icon	Command	Does this
	Quick Connect	Starts the database connection wizard.
	Toggle Browser	Toggles the Browser window on and off.
	Toggle Results	Toggles the Results/Messages window on and off.
	Execute Query	Executes currently selected SQL statement. If script contains multiple statements and none 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.

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

When you switch to DB Query View, 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.

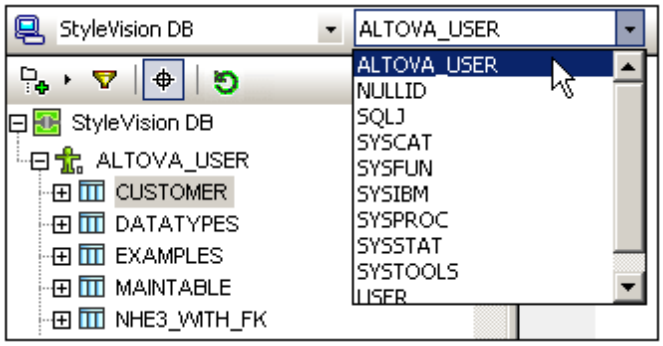


For a list of supported databases, see Database Query | Supported databases.

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.

Browser Pane

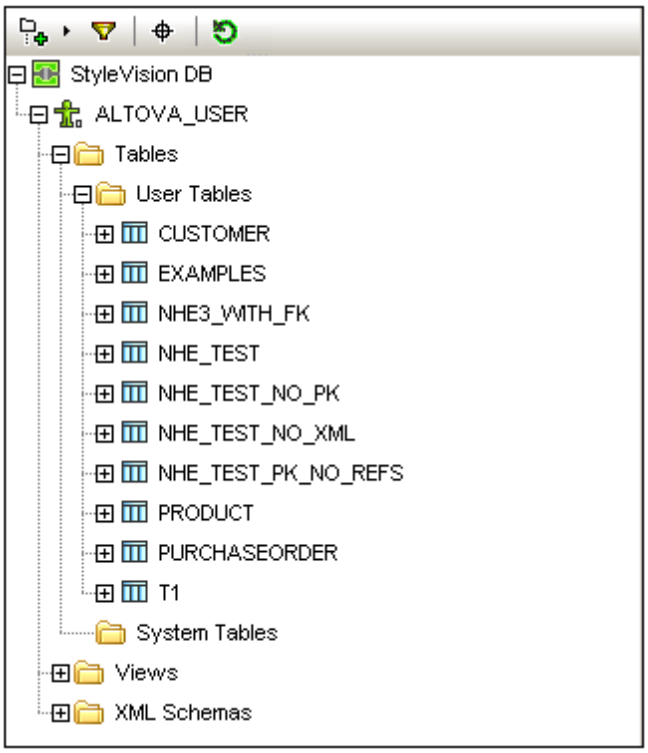
The Browser pane provides an overview of objects in the selected database. This overview includes database constraint information, such as whether a column is a primary or foreign key. In IBM DB2 version 9 and higher databases, the Browser additionally shows registered XML schemas in a separate folder (*see screenshot below*).

This section describes the following:

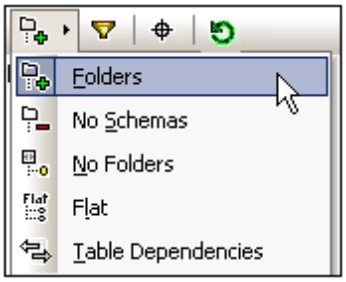
- The layouts available in the Browser pane.
- How to filter database objects.
- How to find database objects.
- How to refresh the root object of the active data source.

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.

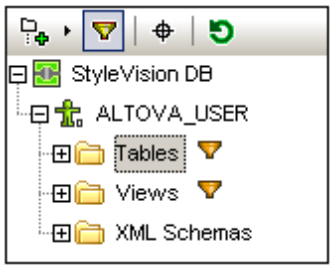
To sort tables into User and System tables, switch to Folders, No Schemas or Flat layout, then right-click the Tables folder and select **Sort into User and System Tables**. The tables are sorted alphabetically in the User Tables and System Tables folders.

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.

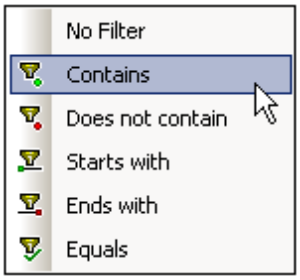
To filter objects in the Browser, do the following:

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*).

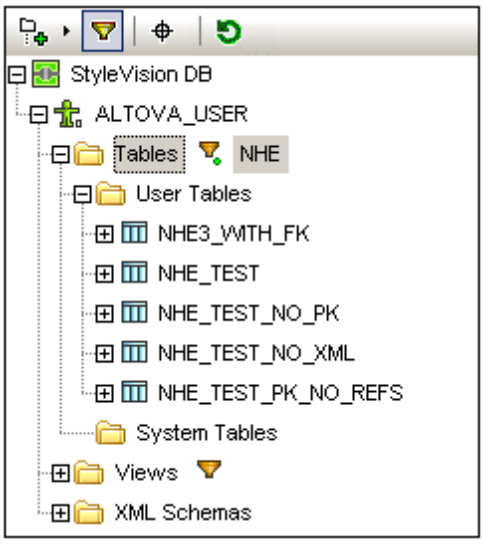


2. Click the filter icon next to the folder you want to filter, and select the filtering option from

the popup menu, for example, `Contains`.



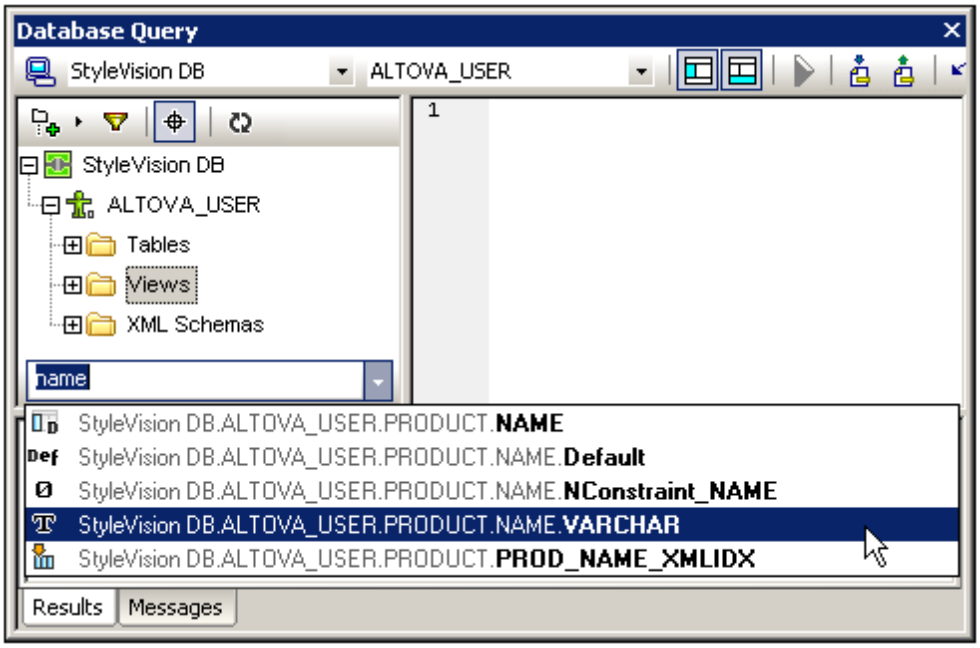
3. In the entry field that appears, enter the filter string (in the screenshot below, the filter string on the `Tables` folder is `NHE`). The filter is applied as you type.



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.



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.

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 of the Database Query window executes the query and displays the result and execution messages in the Results/Messages pane. How to work with queries is described in the next section, Query Pane: Working with Queries. In this section, we describe the main features of the Query pane:

- SQL Editor icons in the Database Query toolbar
- SQL Editor options
- Auto-completion of SQL statements
- Definition of regions in an SQL script
- Insertion of comments in an SQL script
- Use of bookmarks

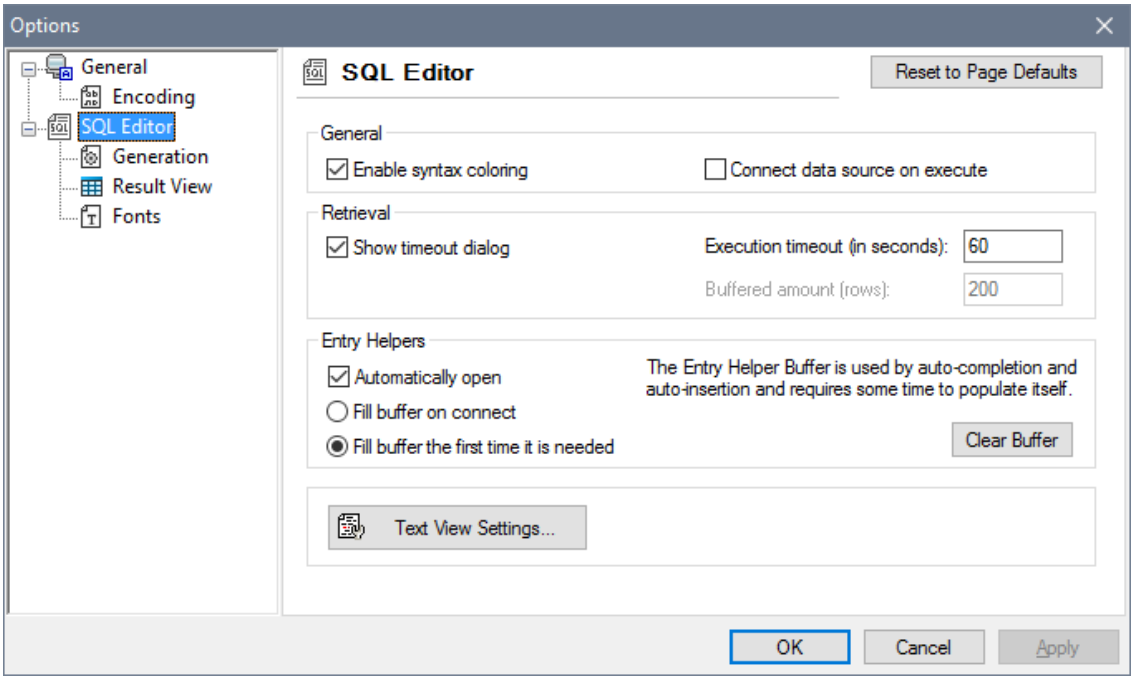
SQL Editor icons in the Database Query toolbar

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*). A page of settings can be selected in the left-hand pane, and 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.

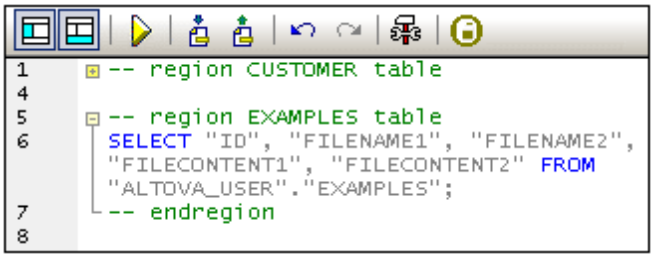


The key settings are as follows:

- ***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 in 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*).



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**.

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. Messages about the execution are displayed in the Messages tab.

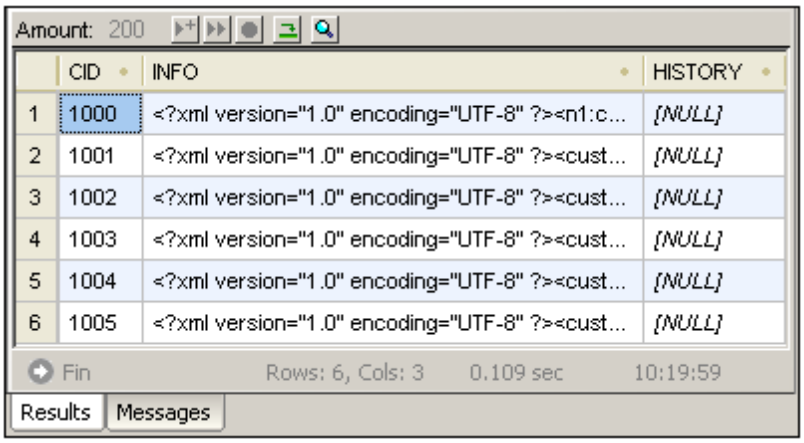
Results and Messages Pane

The Results/Messages pane has two tabs:

- The Results tab shows the data that is retrieved by the query.
- The Messages tab 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*).



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:

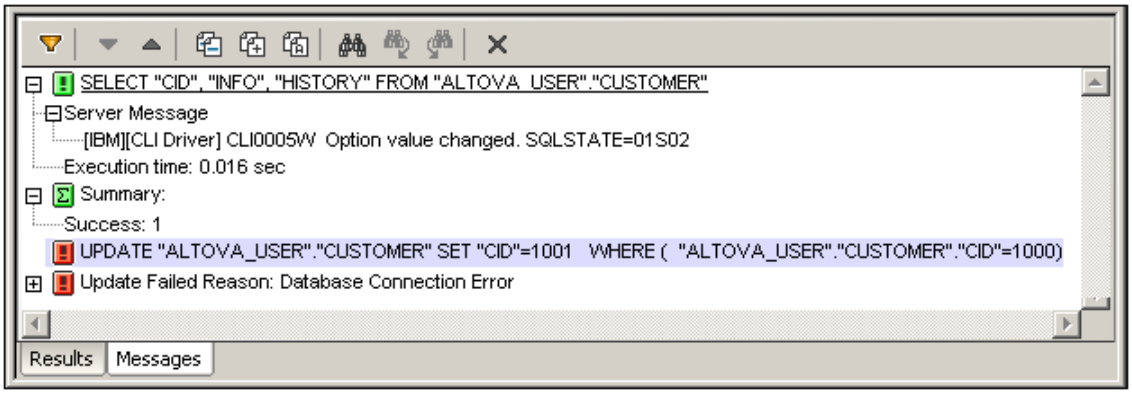
- *Sorting on a column:* Right-click anywhere in the column on which the records are to be sorted, then select **Sorting | Ascending/Descending/Restore Default**.
- *Copying to the clipboard:* This consists of two steps: (i) selecting the data range; and (ii) copying the selection. Data can be selected in several ways: (i) by clicking a column header or row number to select the column or row, respectively; (ii) selecting individual cells (use the **Shift** and/or **Ctrl** keys to select multiple cells); (iii) 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.



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.

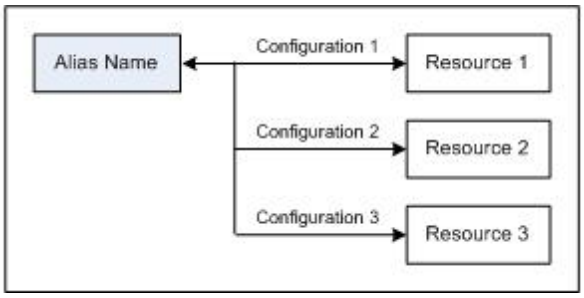
Note: These toolbar icon commands are also available as context menu commands.

Chapter 14

Altova Global Resources

14 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: Resources are defined and the definitions are stored in an XML file. These resources can be shared across multiple Altova applications.
- Using Global Resources: 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.

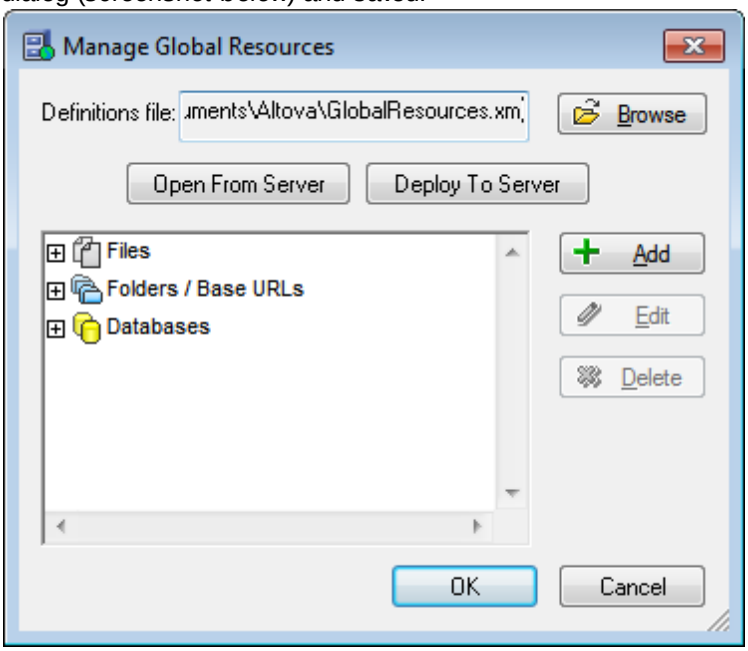
14.1 Defining Global Resources

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.



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, folders, and databases 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.

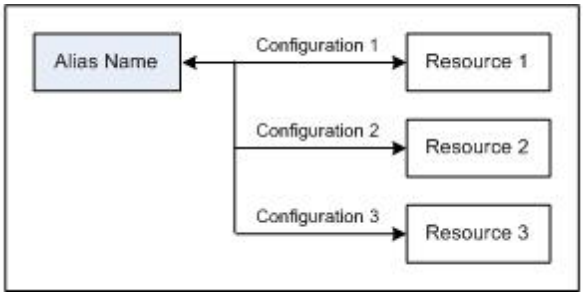
To ***edit a global resource***, select it and click **Edit**. This pops up the relevant **Global Resource** dialog, in which you can make the necessary changes (*see the descriptions of files, folders, and databases in the sub-sections of this section*).

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.

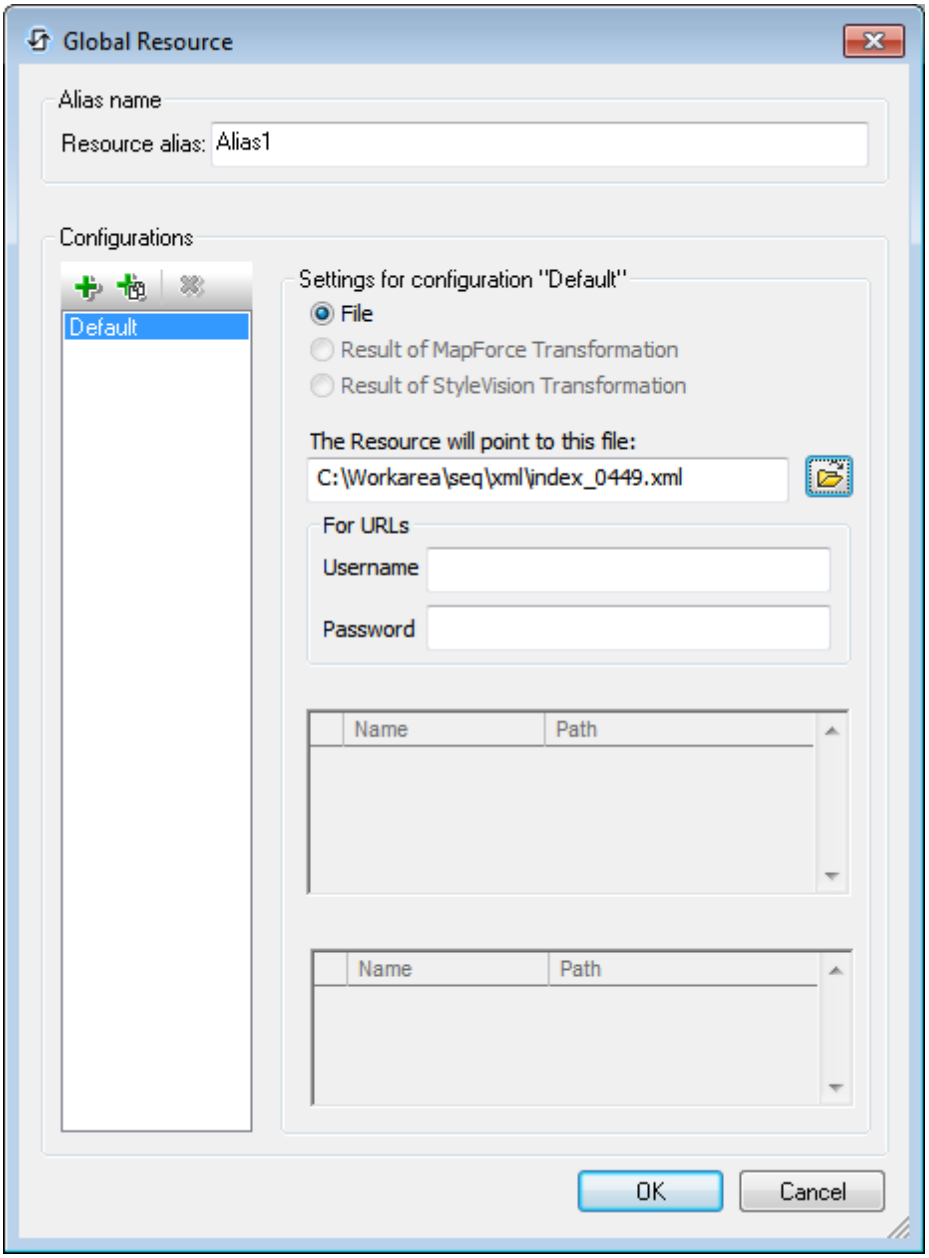
Files

The Global Resource dialog for Files (*screenshot below*) is accessed via the **Add | Files** command in the Manage Global Resources dialog. 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 | Files** command in the Manage Global Resources dialog).

Global Resource dialog

An alias is defined in the Global Resource dialog (*screenshot below*).



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.

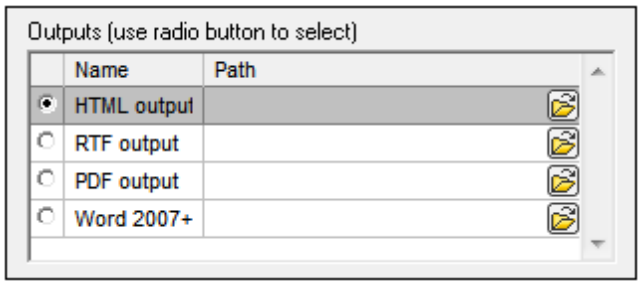
Delete: Deletes the selected configuration.

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:* Select a configuration from the Configurations list, and, in the *Settings for Configuration* pane, specify a resource for the configuration: (i) File, (ii) Output of an Altova MapForce transformation, or (iii) Output of an Altova StyleVision transformation. Select the appropriate radio button. If a MapForce or StyleVision transformation option is selected, then a transformation is carried out by MapForce or StyleVision using, respectively, the `.mfd` or `.sps` file and the respective input file. The result of the transformation will be the resource.
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. If the resource is the result of a transformation, in the *File Selection* text box, browse for the `.mfd` file (for MapForce transformations) or the `.sps` file (for StyleVision transformations). Where multiple inputs or outputs for the transformation are possible, a selection of the options will be presented. For example, the output options of a StyleVision transformation are displayed according to what edition of StyleVision is installed (*the screenshot below shows the outputs for Enterprise Edition*).



Select the radio button of the desired option (in the screenshot above, 'HTML output' is selected). If the resource is the result of a transformation, then the output can be saved as a file or itself as a global resource. Click the icon and select, respectively, Global Resource (for saving the output as a global resource) or Browse (for saving the output as a file). If neither of these two saving options is selected, the transformation result will be loaded as a temporary file when the global resource is invoked.

5. *Define multiple configurations if required:* You can add more configurations and specify a

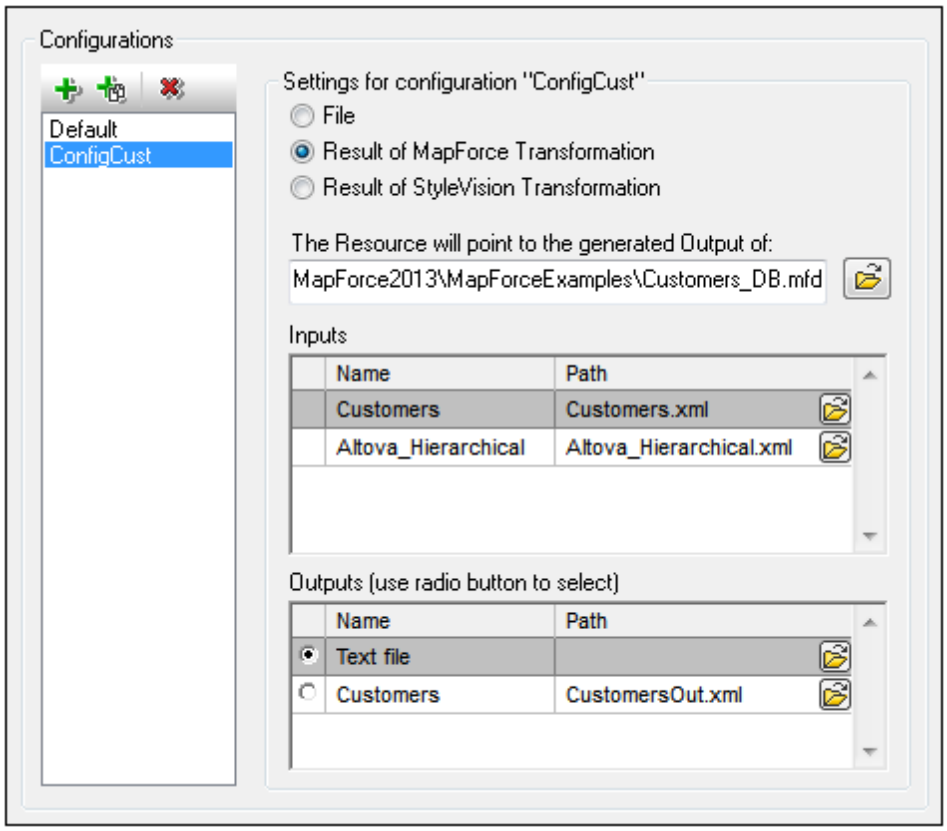
resource for each. Do this by repeating Steps 3 and 4 above for each configuration. You can add a new configuration to the alias definition at any time.

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.

Result of MapForce transformation

Altova MapForce maps one or more (existing) input document schemas to one or more (new) output document schemas. This mapping, which is created by a MapForce user, is known as a MapForce Design (MFD). XML files, text files, databases, etc, that correspond to the input schema/s can be used as data sources. MapForce generates output data files that correspond to the output document schema. This output document is the *Result of MapForce Transformation* file that will become a global resource.

If you wish to set a MapForce-generated data file as a global resource, the following must be specified in the Global Resource dialog (*see screenshot below*):



- ***A `.mfd` (MapForce Design) file.*** You must specify this file in the *Resource will point to generated output of* text box (*see screenshot above*).
- ***One or more input data files.*** After the MFD file has been specified, it is analyzed and, based on the input schema information in it, default data file/s are entered in the *Inputs* pane (*see screenshot above*). You can modify the default file selection for each input

schema by specifying another file.

- ***An output file.*** If the MFD document has multiple output schemas, all these are listed in the *Outputs* pane (*see screenshot above*) and you must select one of them. If the output file location of an individual output schema is specified in the MFD document, then this file location is entered for that output schema in the *Outputs* pane. From the screenshot above we can see that the MFD document specifies that the `Customers` output schema has a default XML data file (`CustomersOut.xml`), while the `Text file` output schema does not have a file association in the MFD file. You can use the default file location in the Outputs pane or specify one yourself. The result of the MapForce transformation will be saved to the file location of the selected output schema. This is the file that will be used as the global resource

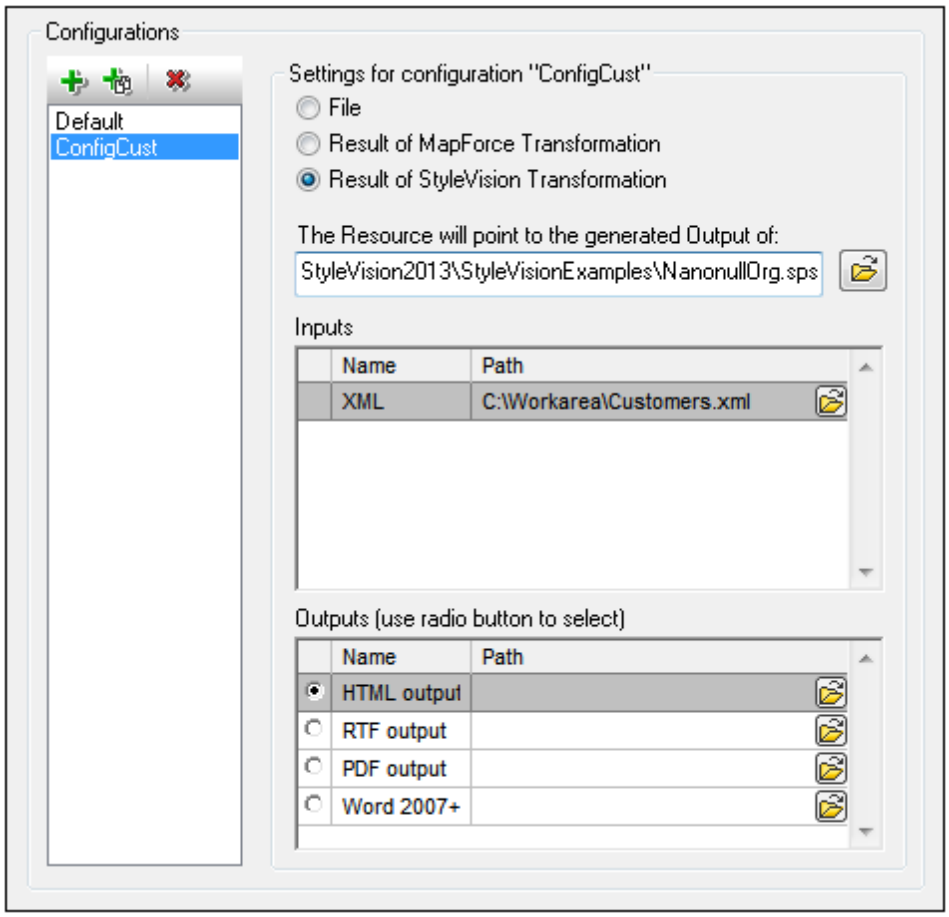
Note: The advantage of this option (Result of MapForce transformation) is that the transformation is carried out at the time the global resource is invoked. This means that the global resource will contain the most up-to-date data (from the input file/s).

Note: Since MapForce is used to run the transformation, you must have Altova MapForce installed for this functionality to work.

Result of StyleVision transformation

Altova StyleVision is used to create StyleVision Power Stylesheet (SPS) files. These SPS files generate XSLT stylesheets that are used to transform XML documents into output documents in various formats (HTML, PDF, RTF, Word 2007+, etc). If you select the option *Result of StyleVision Transformation*, the output document created by StyleVision will be the global resource associated with the selected configuration.

For the *StyleVision Transformation* option in the Global Resource dialog (*see screenshot below*), the following files must be specified.



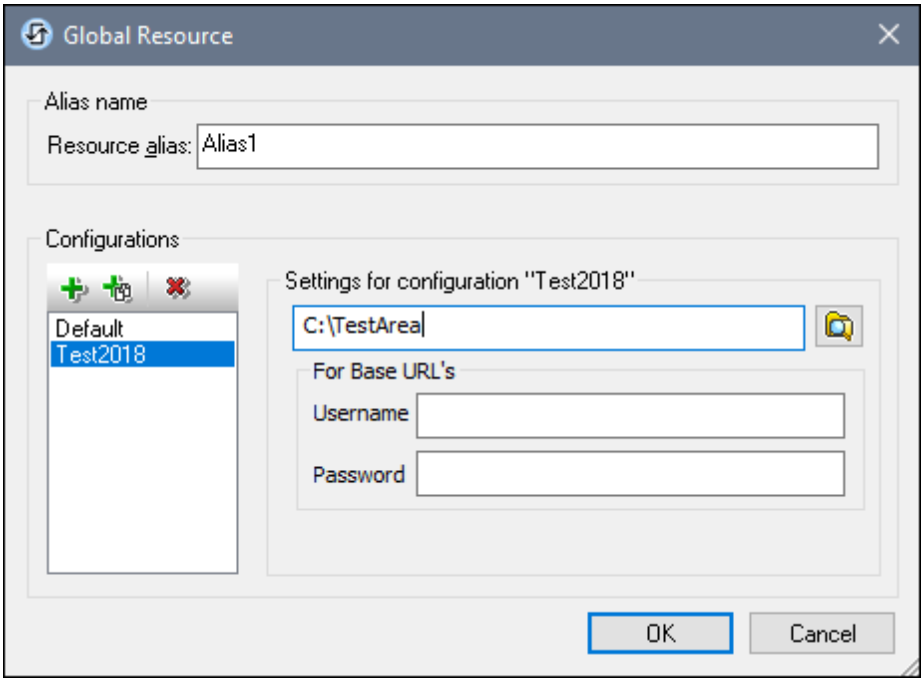
- ***A .sps (SPS) file.*** You must specify this file in the *Resource will point to generated output of* text box (*see screenshot above*).
- ***Input file/s.*** The input file might already be specified in the SPS file. If it is, it will appear automatically in the *Inputs* pane once the SPS file is selected. You can change this entry. If there is no entry, you must add one.
- ***Output file/s.*** Select the output format in the *Outputs* pane, and specify an output file location for that format.

Note: The advantage of this option (Result of StyleVision transformation) is that the transformation is carried out when the global resource is invoked. This means that the global resource will contain the most up-to-date data (from the input file/s).

Note: Since StyleVision is used to run the transformation, you must have Altova StyleVision installed for this functionality to work.

Folders

In the Global Resource dialog for Folders (*screenshot below*), add a folder resource as described below.



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:

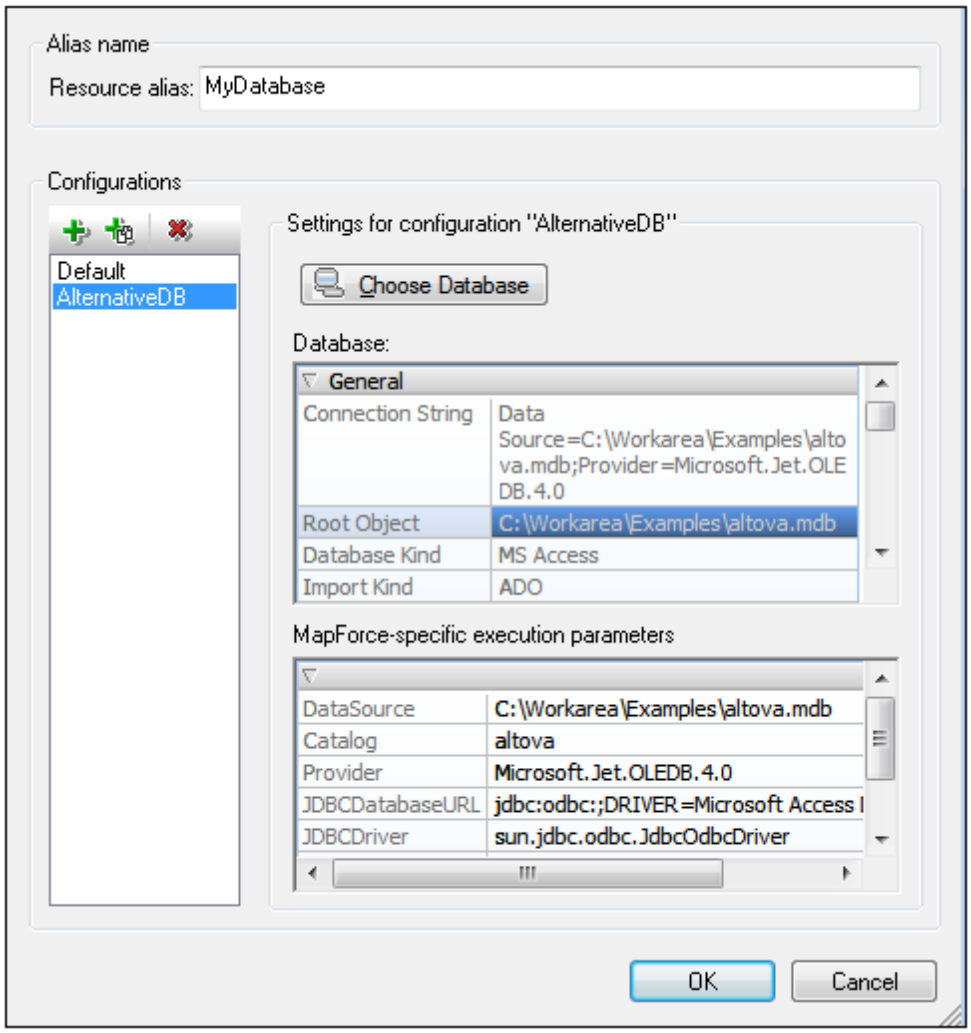
1. *Give the alias a name:* Enter the alias name in the *Resource Alias* text box.
2. *Add configurations:* The Configurations pane will have a configuration named Default (*see screenshot above*). This Default configuration cannot be deleted nor have its name

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.

Databases

In the Global Resource dialog for Databases (*screenshot below*), you can add a database resource as follows:

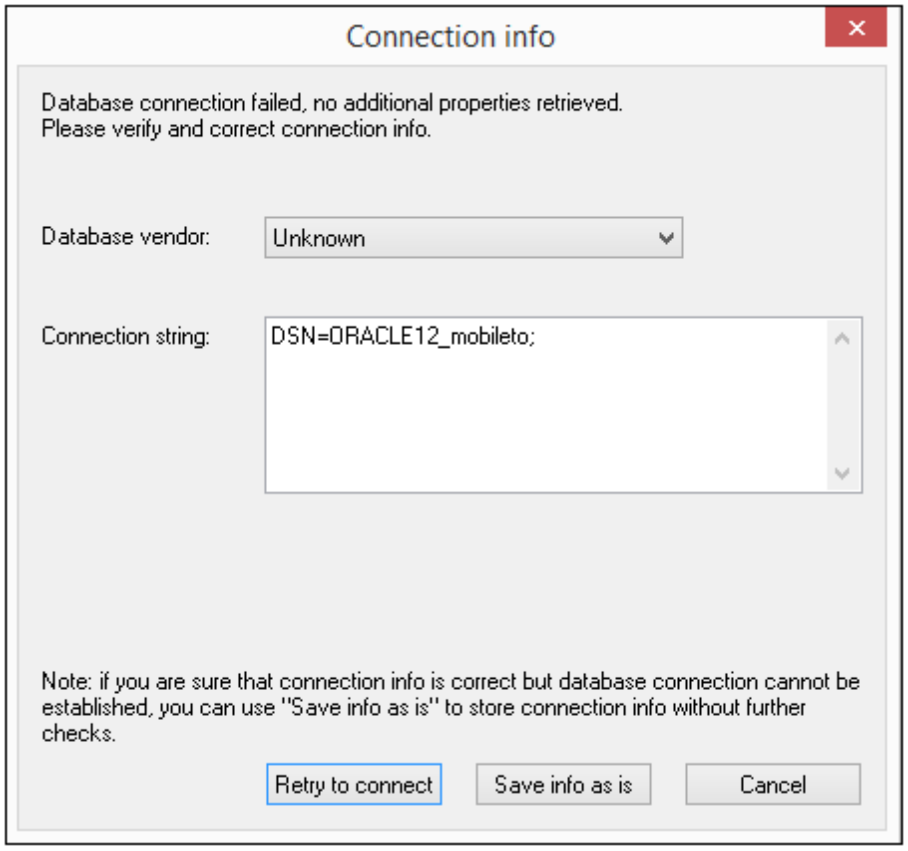


Global Resource dialog icons

- *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.



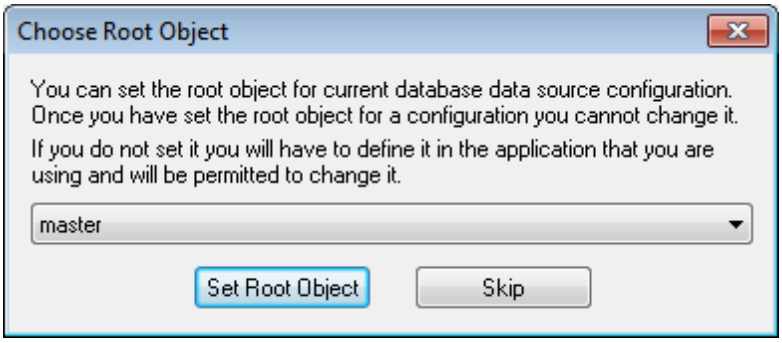
You now have the following options:

- *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.



   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.

14.2 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 and Databases.

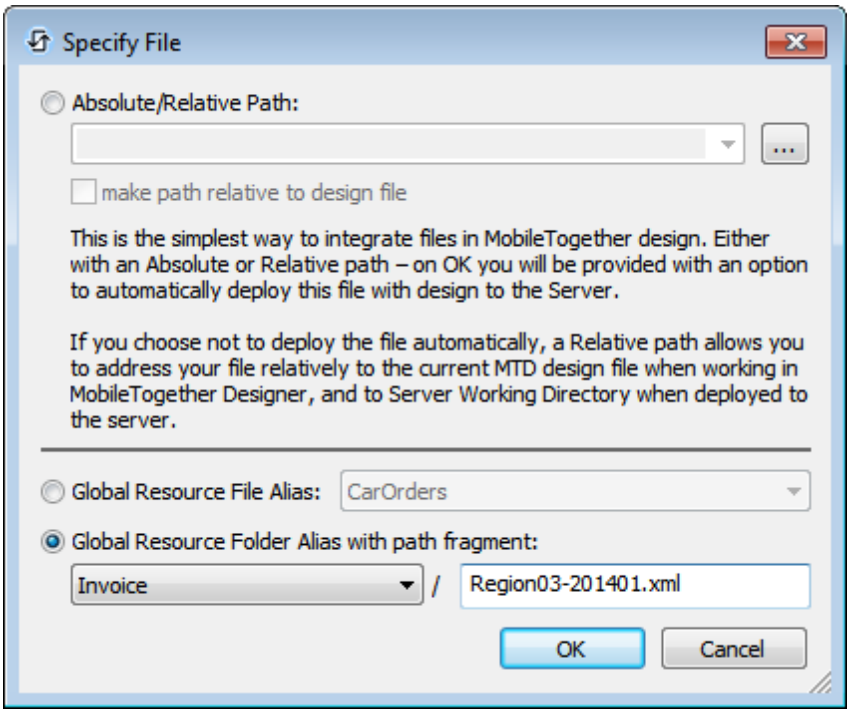
Selections that determine which resource is used

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. 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** 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.

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, the default file of a data source can be assigned via a global resource. In such scenarios, the Specify File (*screenshot below*) appears.

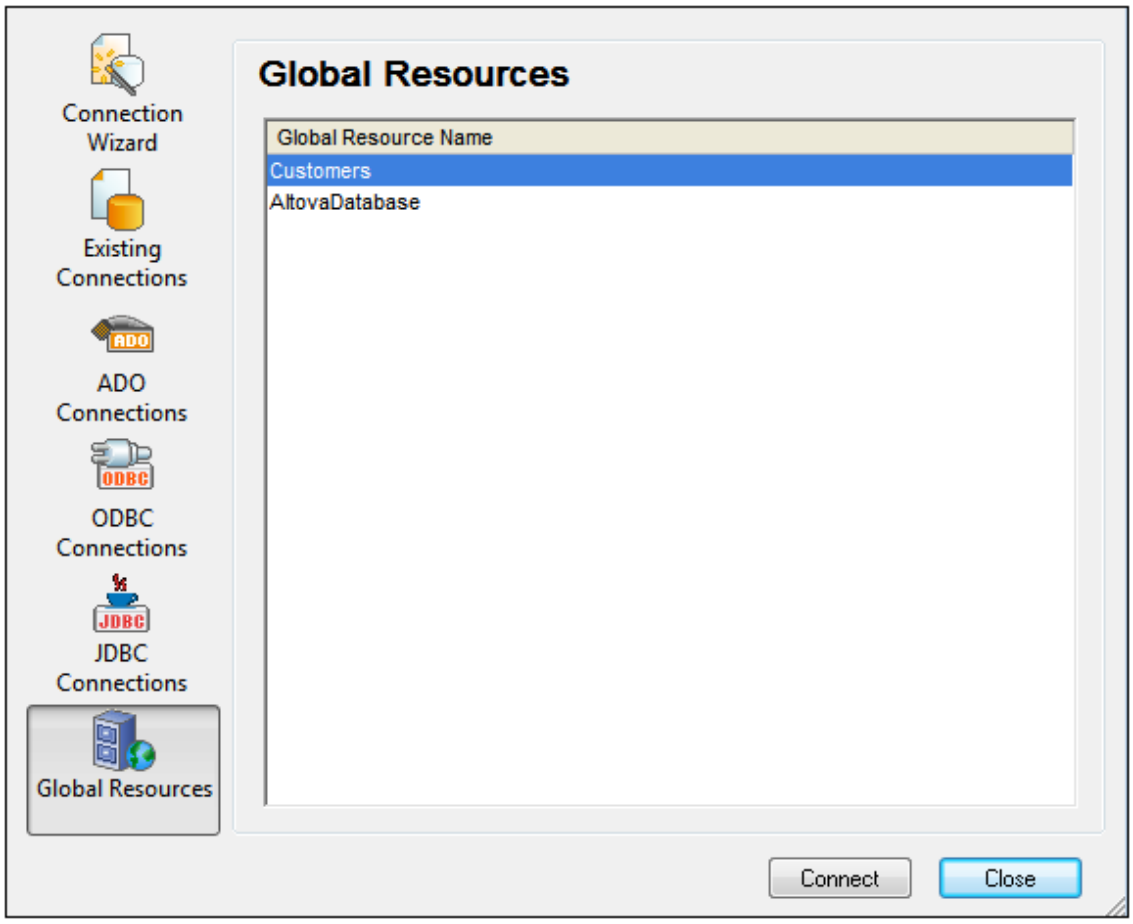


Select whether you wish to use a file-type or folder-type global resource. The combo boxes display, respectively, all the file-type global resources and all the folder-type global resources that have been defined in the currently active Global Resources XML File are. Select the required global resource. 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).

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 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.

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.

Chapter 15

Simulation

15 Simulation

The following simulation methods are available for running and testing a solution:

- Simulation in MobileTogether Designer: A quick way to test whether the design is free from errors.
- Simulation on Server: 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: 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 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:

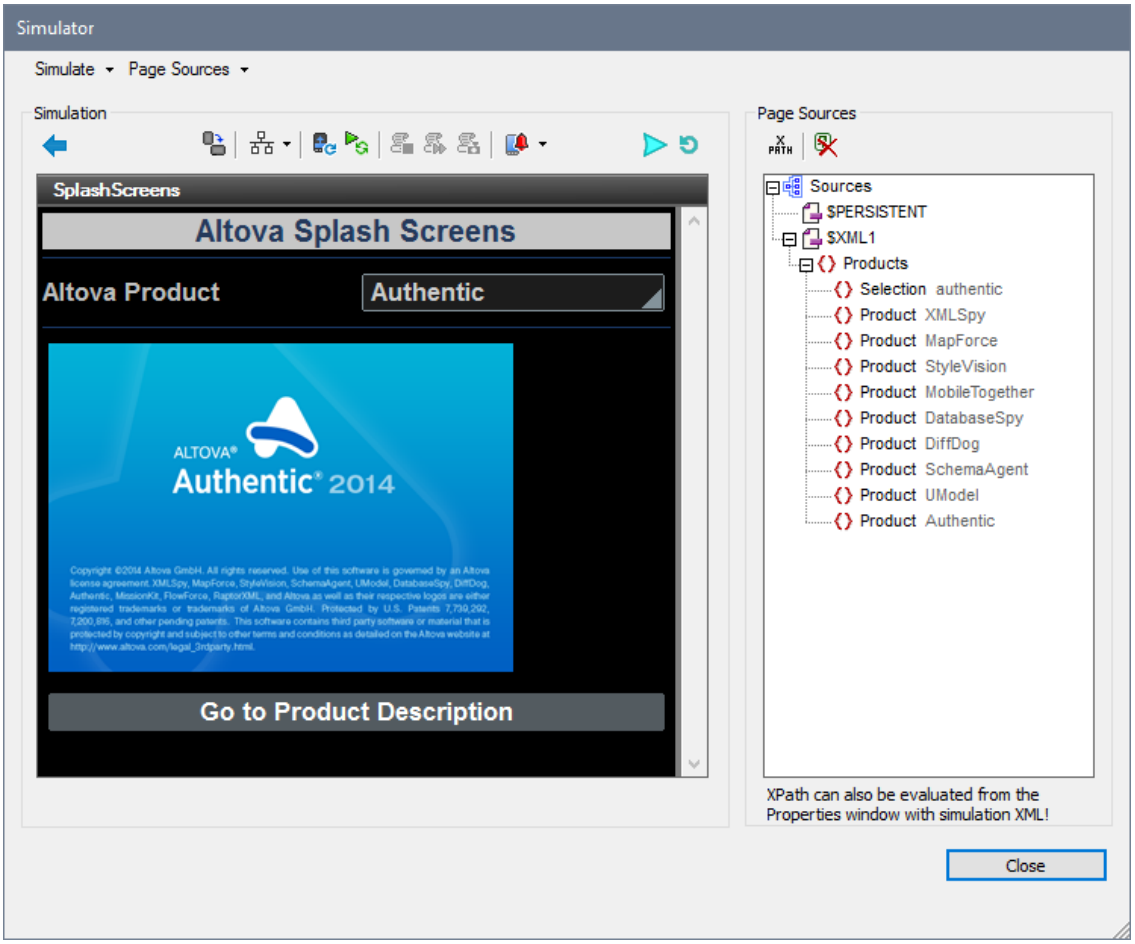
- If you run a designer or server simulation of a design containing geolocation actions, then you will need to define the geolocations to use in the simulation. How to do this is described in the section Geolocation Settings.
- Since the simulators, which run on desktops, are not NFC-enabled, an NFC sample file can be used to simulate an NFC tag. See the section NFC Sample Files 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 for how to create and use such a file.

Simulation language

The simulation language for designer and server simulations is selected via the **Project | Simulation Language** command. The language of client simulations is the same as the language of the client mobile device on which the simulation runs.

15.1 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 **Project | 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.



Simulation language

The simulation language for designer and server simulations is selected via the **Project | Simulation Language** command. The language of client simulations 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.

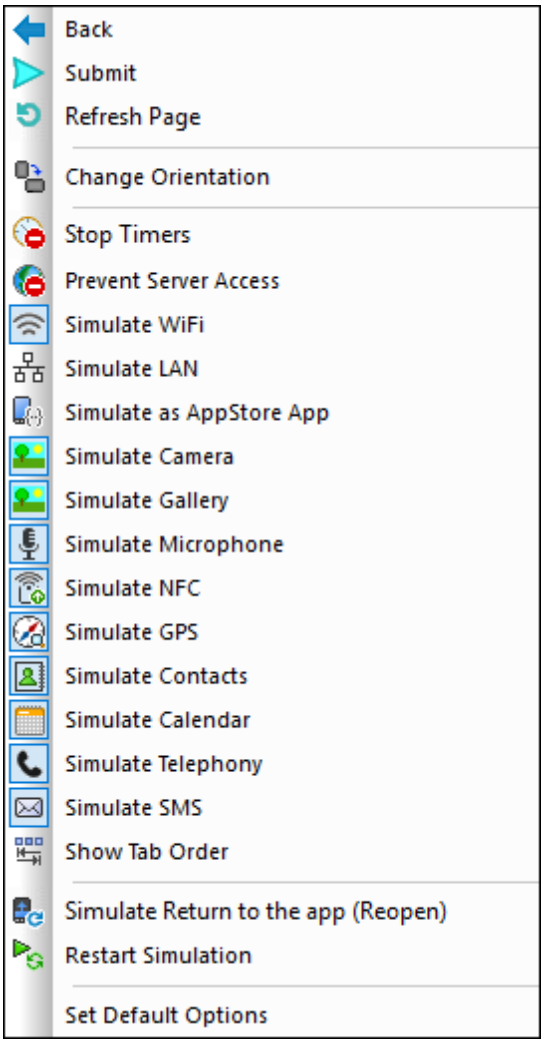
Simulator features

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 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 options

These options are available in the **Simulate** menu (*screenshot below*) and in the Simulate toolbar icon  .



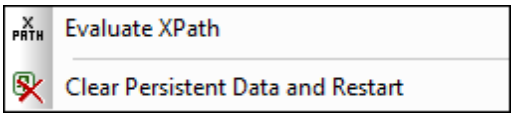
- *Back:* If the page is a subpage, then clicking **Back** closes the subpage. If the page is a top page, the simulator closes. Also see **`OnBackButtonClicked`**.
- *Submit:* If the page is not the last page, then clicking **Submit** takes you to the next page. If the page is the last page, then the workflow is exited. Also see **`OnSubmitButtonClicked`**.
- *Refresh Page:* If underlying data has changed, then **Refresh Page** becomes enabled. On clicking **Refresh Page**, the page is updated with the changed data.
- *Change Orientation:* You can switch the view between landscape and portrait.
- *Stop Timers:* If a timer has been set to run at intervals 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 action, and will allow you to more easily analyze other messages and aspects of the workflow.
- *Prevent Server Access:* Disables access to the server, and so allows you to test the solution's behavior in a server-connection-error scenario. Once clicked, the button toggles to an **Enable Server Access** button. For more information about this feature, see Server Connection Errors.
- *Simulate WiFi:* Sets the **`mt-connected-via-wifi`** 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`** 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`** 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:* Sets the **`mt-nfc-started`** XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This indicates to the simulator that NFC is enabled and that NFC actions can be executed. Actual NFC data is supplied to the simulator via NFC sample files.
- *Simulate GPS:* Sets the **`mt-geolocation-started`** XPath extension function to `true()` when toggled on, and to `false()` when toggled off. The simulator receives an indication that the GPS locator of the device has been started, and it will run the solution as if the geolocation functionality is available. Dummy geolocations can be supplied via the Geolocations XML file, 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 or from your Microsoft Outlook contacts. Which option to use is specified in the Simulation tab of the Options dialog.
- *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 or from your Microsoft Outlook calendar. Which option to use is specified in the Simulation tab of the Options dialog.
- *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.
- *Show Tab Order:* When toggled on, controls that have been defined to be part of a tab order sequence are marked with a blue circle containing a number. The number indicates the control's position in the tab order sequence.
- *Simulate Return to App (Reopen):* Enabled when page refreshes have been defined to occur on page reopens. Refreshes the page during simulations.
- *Restart Simulation:* Restarts the simulation at any time.
- *Set Default Options:* Resets Simulation pane options to their default settings.

Page Sources pane options and features

These options are available in the the **Page Sources** menu and the Page Sources toolbar..



- Clicking **Evaluate XPath** opens the Edit XPath/XQuery Expression dialog, in which you can evaluate XPath expressions. XPath expressions can also be evaluated from the Styles & Properties Pane while the simulation is running.
- Clicking **Clear Persistent Data and Restart** clears persistent data and restarts the simulation.
- 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.
- 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. (The editing commands are available in the context menu of the XML tree.) The Simulation pane will instantly show the modified data. This enables you also 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:

- **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 the structure of the XML tree in the simulator.

15.2 Simulation on Server

A server simulation uses MobileTogether Server to run the simulation (**Project | Use Server for Workflow Simulation**). It reports client messages in the Messages Pane. 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:

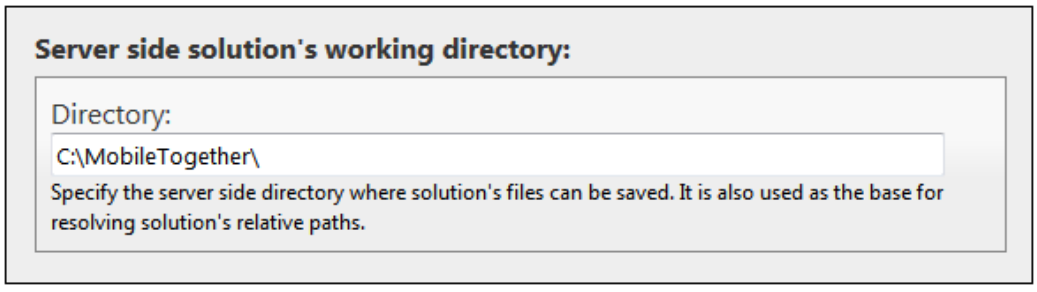
1. The workflow of the active design file in MobileTogether Designer is temporarily passed to MobileTogether Server. So the design file does not need to be deployed to the server in order to see how the design will work from the server.
2. The server serves the workflow to the simulator of MobileTogether Designer. In this way the simulator plays the role of a client.

Simulation language

The simulation language for designer and server simulations is selected via the **Project | Simulation Language** command. The language of client simulations is the same as the language of the client mobile device on which the simulation runs.

Running the simulation

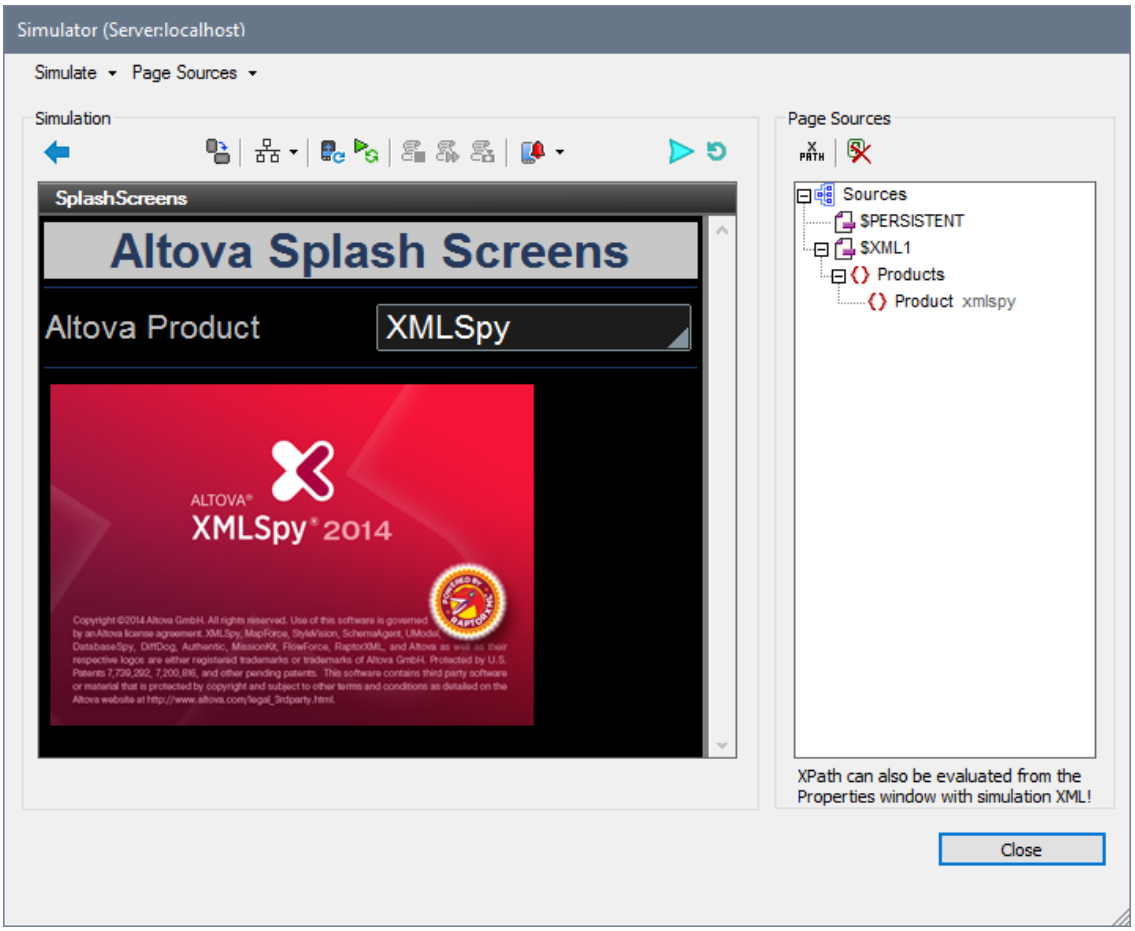
1. Start MobileTogether Server. See the MobileTogether Server user manual for information about how to do this.
2. In the Web UI of MobileTogether Server, you must set the solution's working directory (**Settings | Server Side Solution's Working Directory**, *see screenshot below*). All relative paths 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.



3. In MobileTogether Designer, make sure that the server settings are correctly defined.
4. In MobileTogether Designer, select **Project | 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.



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.

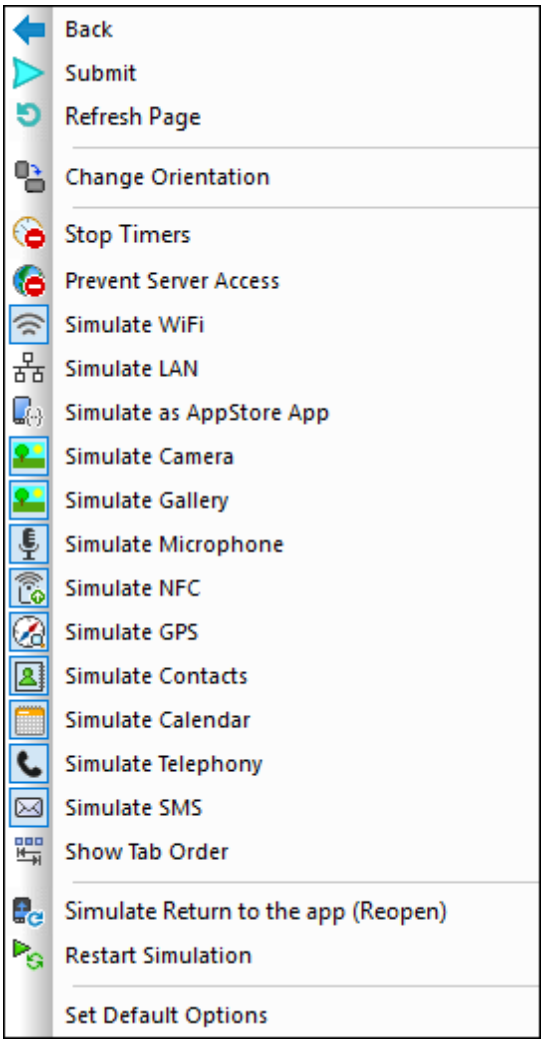
Simulator features

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 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 options

These options are available in the **Simulate** menu (*screenshot below*) and in the Simulate toolbar icon  .



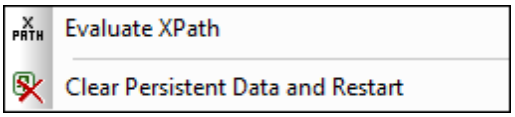
- *Back:* If the page is a subpage, then clicking **Back** closes the subpage. If the page is a top page, the simulator closes. Also see **`OnBackButtonClicked`**.
- *Submit:* If the page is not the last page, then clicking **Submit** takes you to the next page. If the page is the last page, then the workflow is exited. Also see **`OnSubmitButtonClicked`**.
- *Refresh Page:* If underlying data has changed, then **Refresh Page** becomes enabled. On clicking **Refresh Page**, the page is updated with the changed data.
- *Change Orientation:* You can switch the view between landscape and portrait.
- *Stop Timers:* If a timer has been set to run at intervals 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 action, and will allow you to more easily analyze other messages and aspects of the workflow.
- *Prevent Server Access:* Disables access to the server, and so allows you to test the solution's behavior in a server-connection-error scenario. Once clicked, the button toggles to an **Enable Server Access** button. For more information about this feature, see Server Connection Errors.
- *Simulate WiFi:* Sets the **`mt-connected-via-wifi`** 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`** 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`** 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:* Sets the **`mt-nfc-started`** XPath extension function to `true()` when toggled on, and to `false()` when toggled off. This indicates to the simulator that NFC is enabled and that NFC actions can be executed. Actual NFC data is supplied to the simulator via NFC sample files.
- *Simulate GPS:* Sets the **`mt-geolocation-started`** XPath extension function to `true()` when toggled on, and to `false()` when toggled off. The simulator receives an indication that the GPS locator of the device has been started, and it will run the solution as if the geolocation functionality is available. Dummy geolocations can be supplied via the Geolocations XML file, 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 or from your Microsoft Outlook contacts. Which option to use is specified in the Simulation tab of the Options dialog.
- *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 or from your Microsoft Outlook calendar. Which option to use is specified in the Simulation tab of the Options dialog.
- *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.
- *Show Tab Order:* When toggled on, controls that have been defined to be part of a tab order sequence are marked with a blue circle containing a number. The number indicates the control's position in the tab order sequence.
- *Simulate Return to App (Reopen):* Enabled when page refreshes have been defined to occur on page reopens. Refreshes the page during simulations.
- *Restart Simulation:* Restarts the simulation at any time.
- *Set Default Options:* Resets Simulation pane options to their default settings.

Page Sources pane options and features

These options are available in the the **Page Sources** menu and the Page Sources toolbar..



- Clicking **Evaluate XPath** opens the Edit XPath/XQuery Expression dialog, in which you can evaluate XPath expressions. XPath expressions can also be evaluated from the Styles & Properties Pane while the simulation is running.
- Clicking **Clear Persistent Data and Restart** clears persistent data and restarts the simulation.
- 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.
- 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. (The editing commands are available in the context menu of the XML tree.) The Simulation pane will instantly show the modified data. This enables you also 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:

- **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 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 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.

15.3 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. Simulation on the client assumes that your mobile device can connect to your PC over wireless LAN. Do a trial run on the client (simulation on the client) as follows:

1. Client: Set up a new server
   1. Start MobileTogether on the client and click the **Settings | Server** icon.
   2. Add a new server and enter all the relevant data (server address, port number (`8083` is the default server port for client access), username, and password).
   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, stop it running as a service.
   2. In MobileTogether Designer, select **Tools | Options**, click the *Trial Run on Client* tab, and enter the PC port number that the client is going to use (`8083` is the default).
   3. Open the project (`.mtd`) file that you want to test on the client.
   4. Select **Project | Trial Run on Client**. This opens a dialog box that will show the page sources and data structure.

3. Client: Run the simulation
   1. Start the MobileTogether Client app on your mobile device if not already started. Only the design files that are currently open on the PC are visible on the mobile client (as solutions) using this type of simulation.
   2. Select the solution that you want to test.
   3. This opens a prompt on the PC asking if you want to start the solution. Click **Yes**. The tree structure now appears in the dialog box of the PC. After a short time, the solution runs on the client.
   4. 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.

Simulation language

The simulation language for designer and server simulations is selected via the **Project | Simulation Language** command. The language of client simulations is the same as the language of the client mobile device on which the simulation runs.

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. (The editing commands are available in the context menu of the XML tree.) The Simulation pane will instantly show the modified data. This enables you also 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:

- **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 the structure of the XML tree in the simulator.

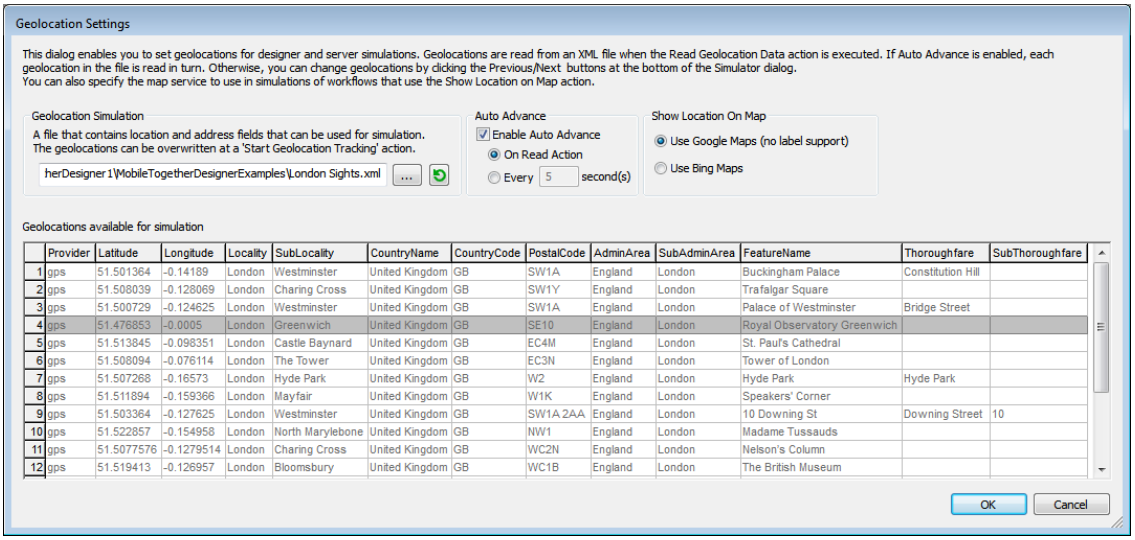
15.4 Geolocation Settings

The Geolocation Settings dialog (*screenshot below*) enables you to specify geolocations from an XML file for designer and server 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 **Project | Simulate Workflow (F5)** or **Project | 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 and server simulations.

Note: If no geolocation action is used in the design, the **Geolocation** button will be **not be displayed** in the Simulator dialog.



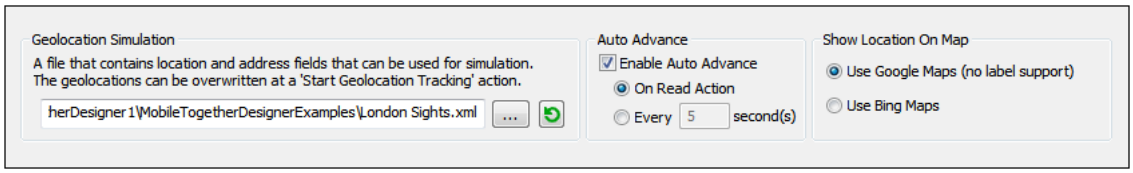
The geolocation settings

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 reads 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 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 data 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.



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 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`, `W/E`)
  `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/W`**) 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`, `W/E`)
  `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/W`**) is optional
  `+/-D°M.MM'  +/-D°M.MM'`
  Example: **`+33°55.55'  -22°44.44'`**

- Decimal degrees, with suffixed orientation (`N/S`, `W/E`)
  `D.DDN/S  D.DDW/E`
  Example: **`33.33N  22.22W`**

- Decimal degrees, with prefixed sign (+/-); the plus sign for (**N/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

15.5 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 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.

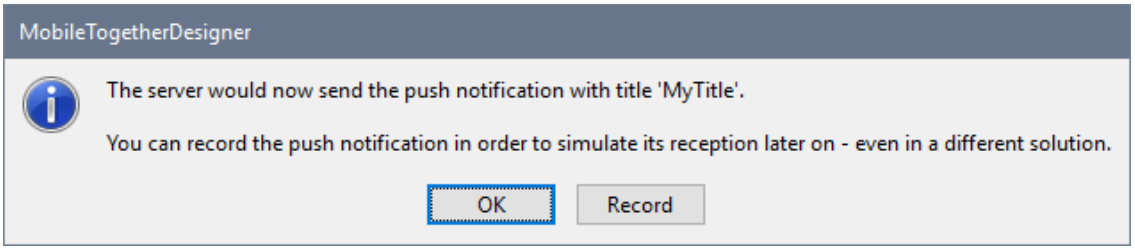
15.6 Push Notification Simulations

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.



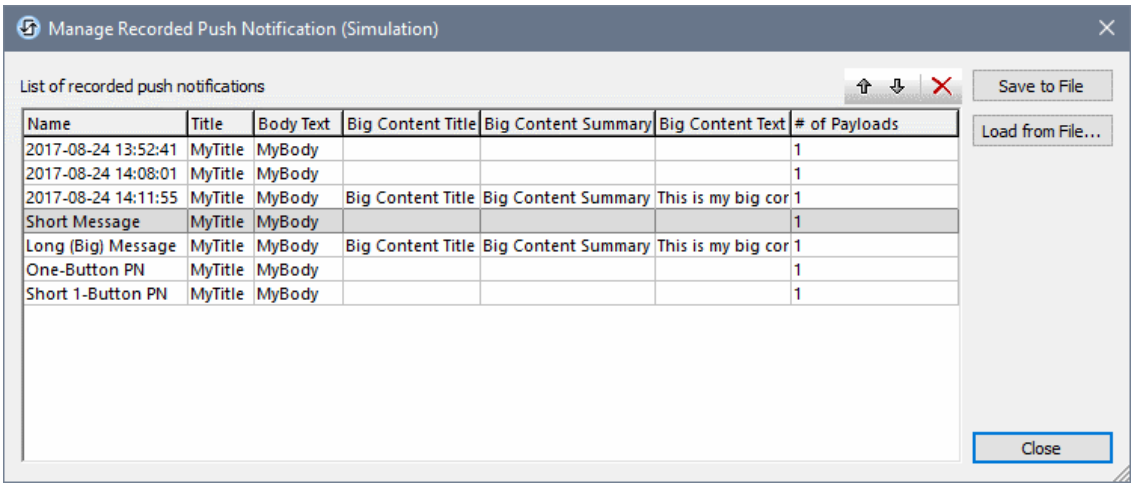
Click **Record** to record the PN data to memory. You can record multiple PNs to memory in this way. On closing the simulation, the recorded PNs are in memory—but not yet saved to file.

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

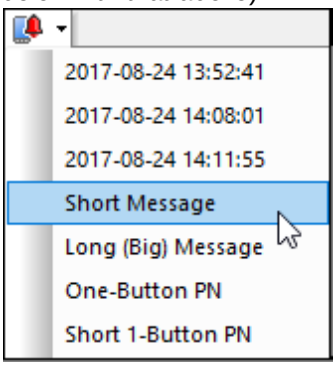
To load recorded PN simulation data from an MT PN Simulation file (`.mtpnsim` file), start a simulation of the receiving solution (an MTD file) and then click the **Push Notifications** icon  in the simulator's toolbar. This causes the Manage Recorded Push Notification (Simulation) dialog (*see screenshot 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.



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.

15.7 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, that this file is to be used for simulations involving the Read Contacts 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.

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>
```

15.8 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 pane of the Options dialog, that this file is to be used for simulations involving the Access Calendar 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.

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="1" Location="Meeting Room 2">
         <Attendee Name="Bob" Status="Accepted" Type="Required"
Relationship="Speaker"/>
      </Event>
      <Event Id="1" 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="1" 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.

15.9 Service Trigger Simulations

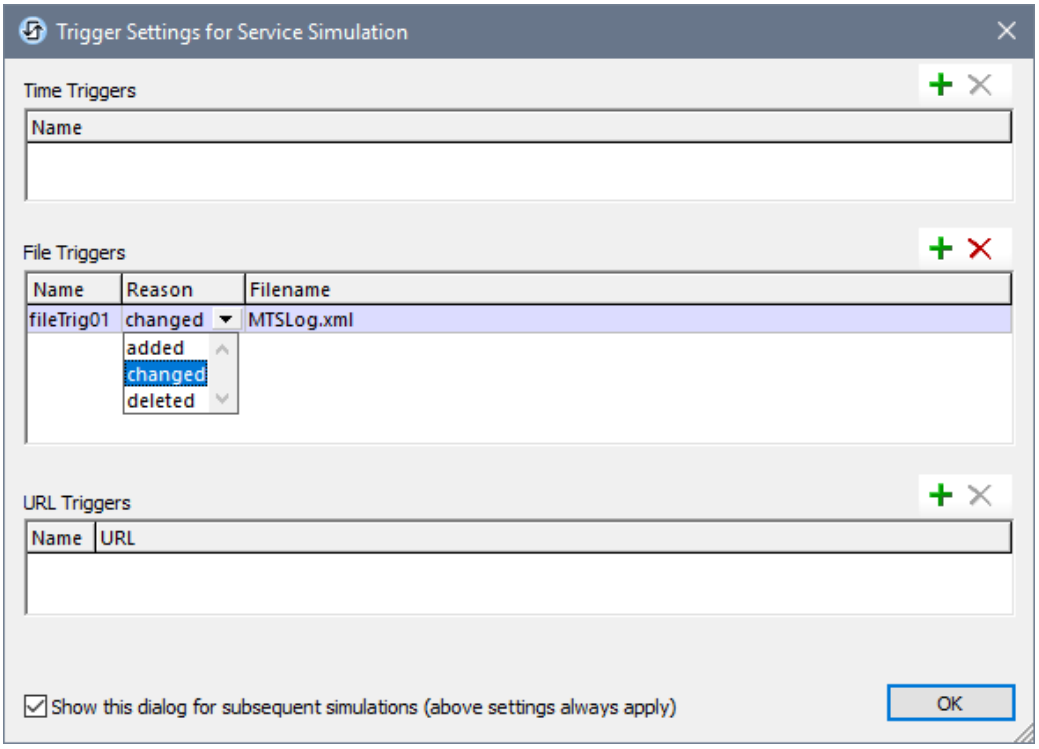
When a server service is simulated, the actions defined for the service are executed. If any of these actions make use of data in the **$MT_SERVICE** page source tree, then, for the simulation, you will need to manually create the data in the tree. This is because in a real-use scenario the data in the **$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 or server. 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, make sure that the names you enter for triggers match the names of triggers on the server.



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**), 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*).

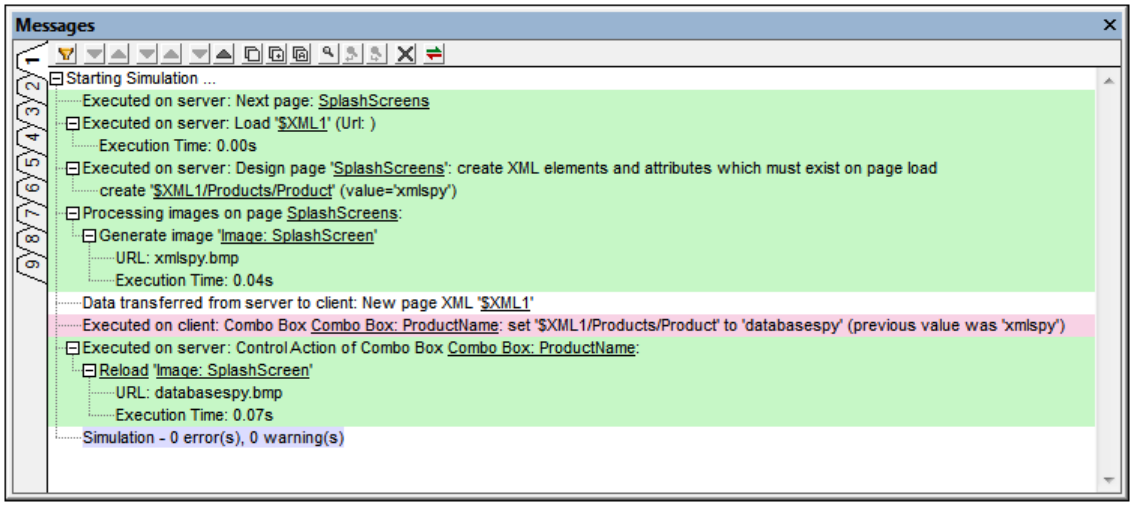
For a detailed description of how to create services, see the section Server Services.

15.10 Messages Pane

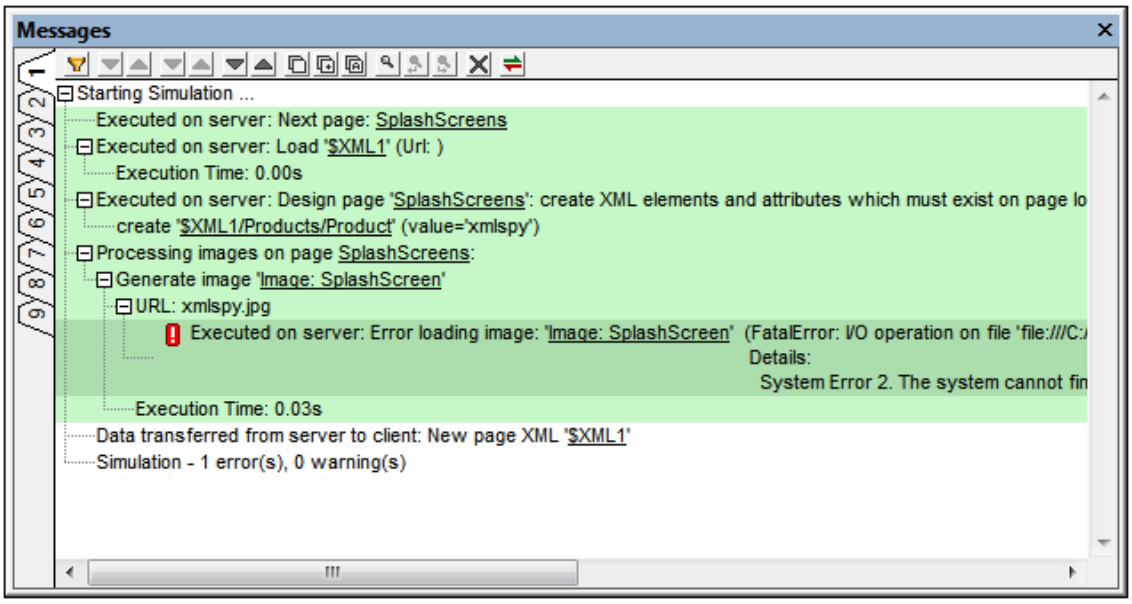
While the simulation progresses, the Messages Pane of the MobileTogether Designer GUI provides a detailed and step-by-step report of all the solution-related activity taking place, and the time taken for each action. You can clearly see what is happening at each step in the progress of the workflow: 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.) Links in the Messages Pane take you to the relevant component in the design or the Page Sources Pane. 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*).

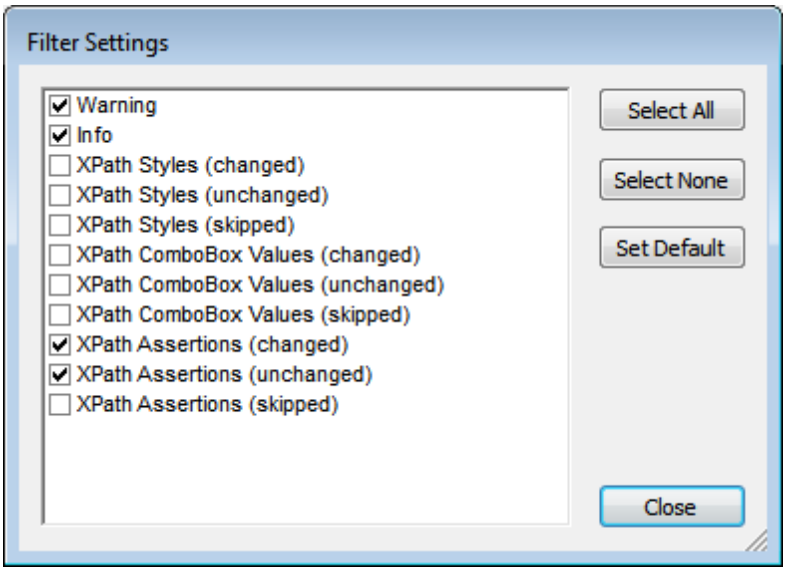


If an error occurs, it is flagged with a red icon (*see screenshot below*).



Filtering messages

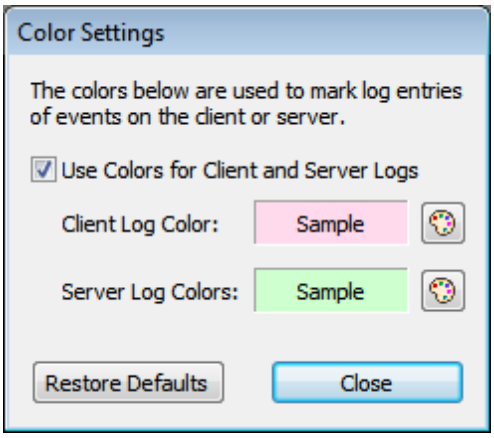
You can specify what kind of messages are displayed in the Messages Pane. To do this, click the **Filter** button in the toolbar of the Messages Pane (*screenshot above*). This displays the Filter Settings dialog (*screenshot below*). Select the message types you want to display and click **Close**. 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.



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.



For more information, see the Messages Pane.

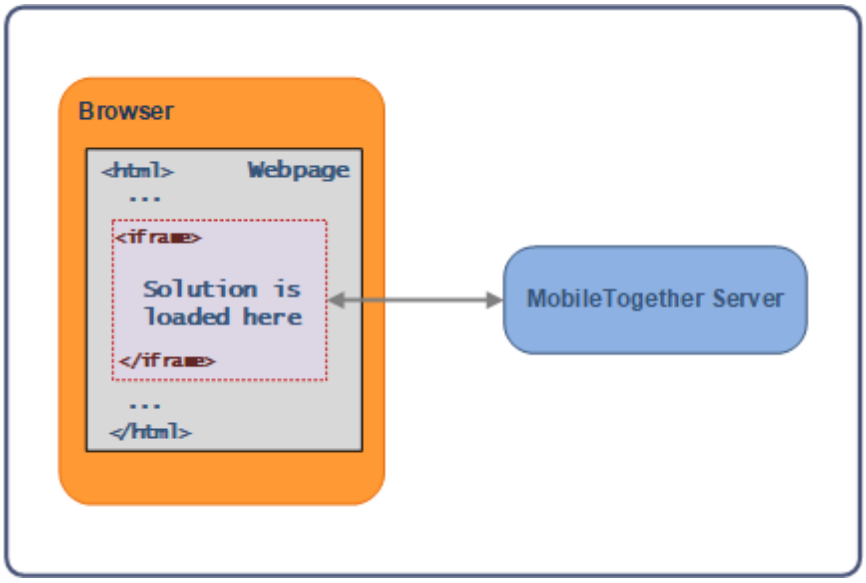
Chapter 16

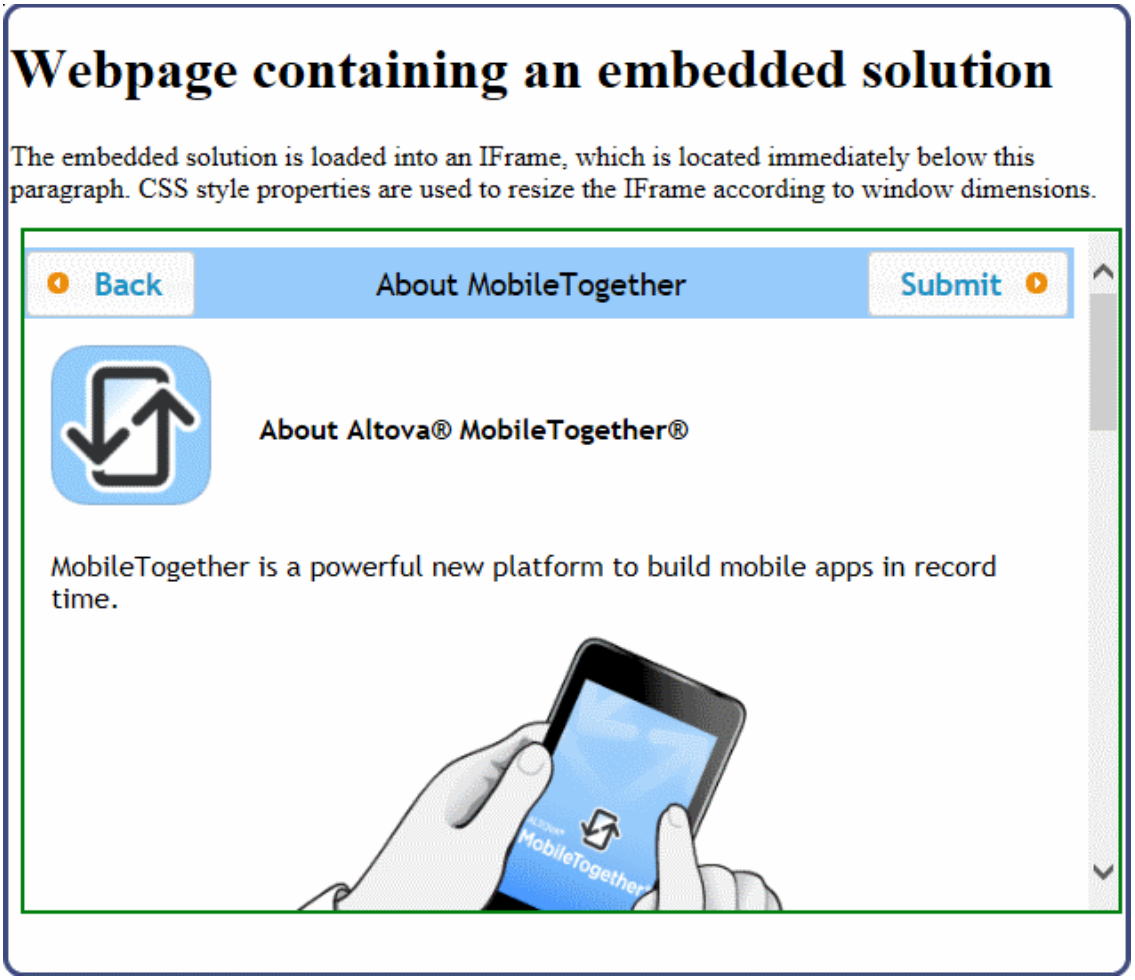
Embedded Webpage Solutions

16 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).





This section is organized as follows:

- Embedding a Solution in a Webpage 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: 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 describes communication from webpage to IFrame to MobileTogether Server, while the section Listening: From Solution to Webpage explains how to communicate from MobileTogether Server to IFrame to webpage.
- Authentication: 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) 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: 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** page event of the solution picks up the message from the webpage
- The **$MT_EMBEDDEDMESSAGE** JSON page source stores the received data in a structured form
- The Load from String 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 action serializes a page source and places the resultant string in the node of another page source
- The Embedded Message Back action sends a serialized JSON string to the IFrame that loaded the current solution

16.1 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:* 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:* 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:* 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 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>
```

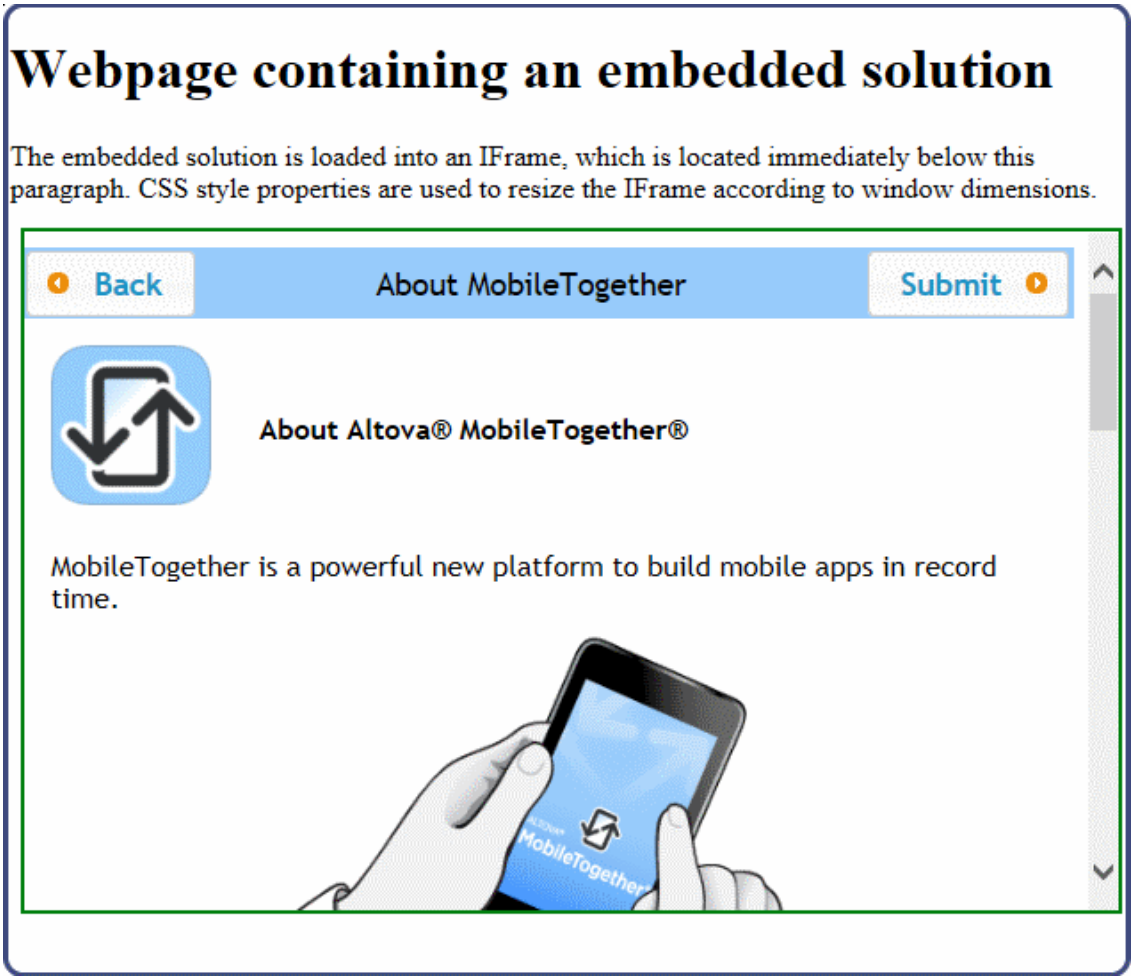
The URL in the **src** attribute must resolve to the following pattern:

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.
Back
About MobileTogether
Submit
About Altova® MobileTogether®
MobileTogether is a powerful new platform to build mobile apps in record time.
MobileTogether

16.2 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
- Listening: From Server to Webpage

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).

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` 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 and XML 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 and XML examples. Also see the description of the `OnEmbeddedMessage` 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(myJSO
NData, '*');
        }
```

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` 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 and XML examples.

Listening: From Server to Webpage

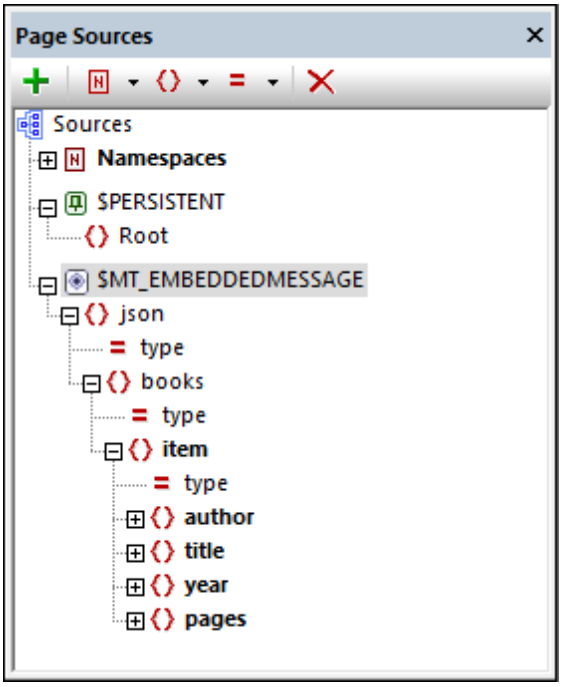
A message is sent from MobileTogether Server to a webpage in the following stages:

1. The workflow's Embedded Message Back 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.

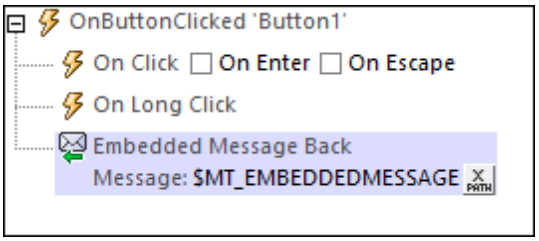
Example

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 action set for its **OnClicked** event (*see screenshot below*). In the XPath expression below, note that it is the contents of **$MT_EMBEDDEDMESSAGE** node that is sent as the message event, that is, the **json** node and its contents are sent.



In the HTML page, we can now register an event listener:

```
window.addEventListener('message', ProcessReturnMsg)
```

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
   /* '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**.

16.3 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:* The user does not need to log in
- *User login:* 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:* 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).

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`).

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:

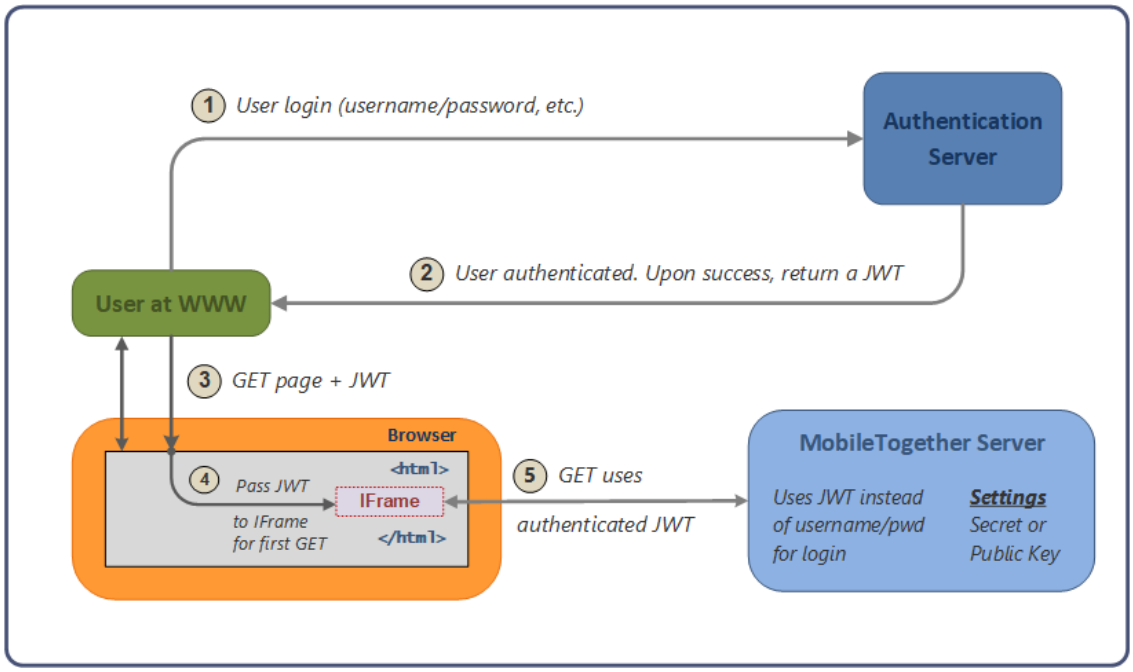
- The user must be a registered user of MobileTogether Server
- The user must have the appropriate permissions to access the targeted solution..

For more information about registering users with MobileTogether Server and setting up their permissions, see the MobileTogether Server documentation.

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.



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 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*.

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 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.

Symmetric Key: a Shared Secret

The procedure for using a JWT in an embedded webpage solution:

1. Create a JWT 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 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.

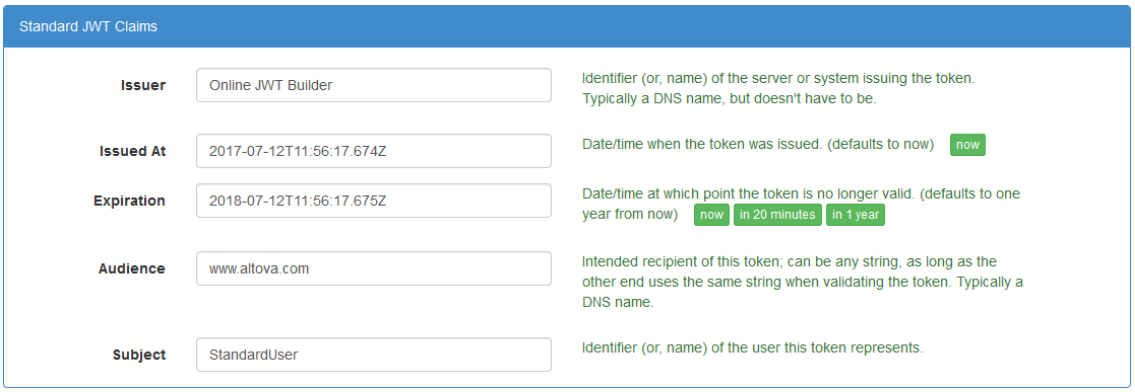
When the JWT is passed to the server, the server validates it by using the *Audience* information and the shared secret that you entered in the settings to generate the JWT.

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

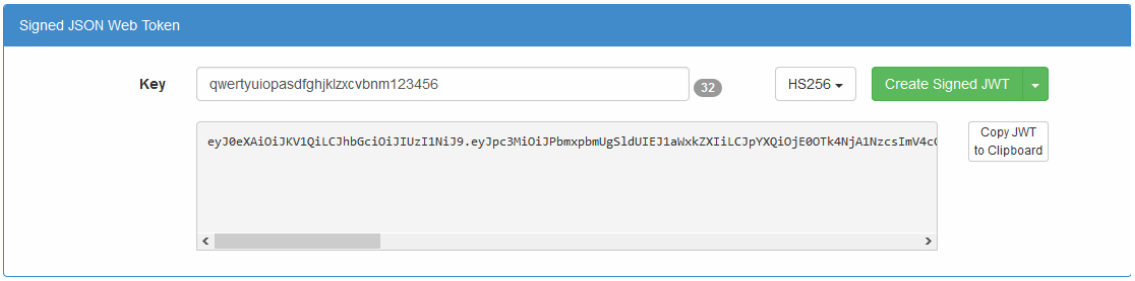
The standard claims *(see screenshot below)* make up the core claims:



- 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 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 below*).

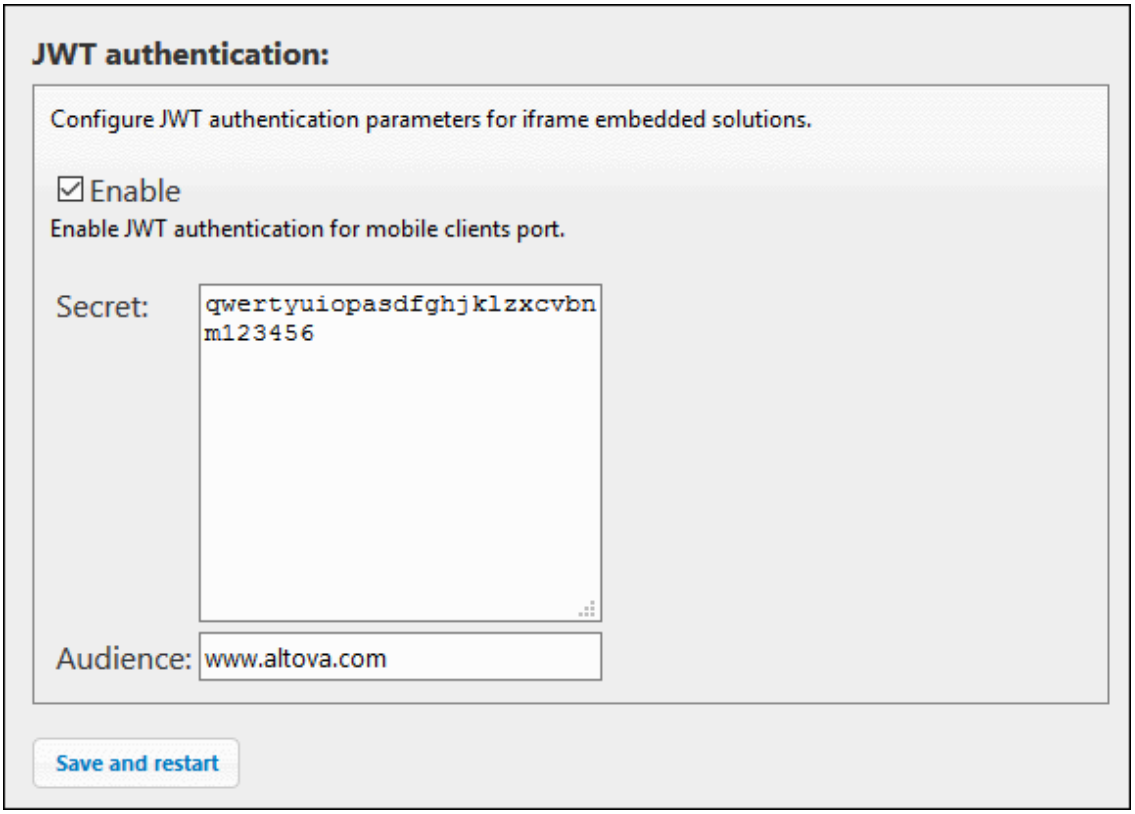
In the screenshot below, a 32-character-long secret is entered, and the encryption algorithm **HS256** is selected. On clicking **Create Signed JWT**, the JWT is created and is displayed in the text box.



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, 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.

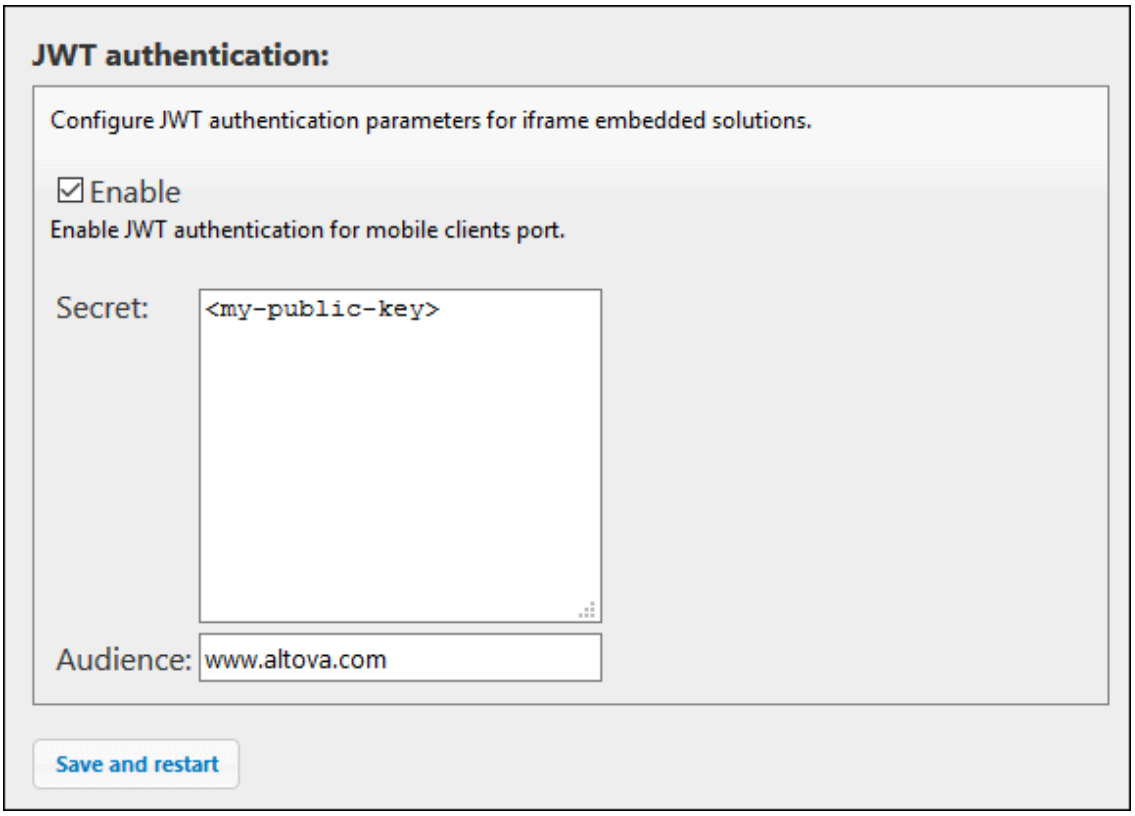

JWT authentication:
Configure JWT authentication parameters for iframe embedded solutions.
Enable
Enable JWT authentication for mobile clients port.
Secret:
qwertyuiopasdfghjklzxcvbnm123456
Audience:
www.altova.com
Save and restart

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, enter the shared secret.)
- *Audience:* Enter the same string as that you entered for this claim when creating the JWT.



16.4 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 *(My) Documents* MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions`**.

In this section:

- Embedding a Solution
- Sending/Receiving JSON Data
- Sending/Receiving XML Data
- Pre-setting the JSON Page Source
- JWT Authentication

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 of the `About` workflow.

See also: Embedding a Solution in a Webpage

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:

Load


```
{
 "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

Save

The embedded webpage solution consists of the HTML webpage (**jsonBooks.html**) and a MobileTogether design (**jsonBooks.mtd**). Both files are located in your *(My) Documents* MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions**. To try out the files, deploy the MTD file to your server and enable it to be accessed anonymously. If needed, modify the HTML code so that the IFrame correctly targets the workflow on the server. 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(book
s, '*');
        }


       // 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:*

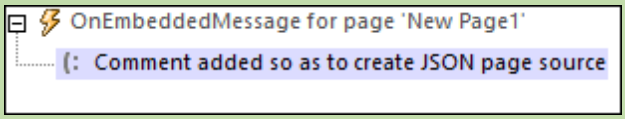
```
<button onclick="sendbooks()">Load</button>
```

A JavaScript function uses **postMessage()** to send the contents of the **books** variable to the IFrame.

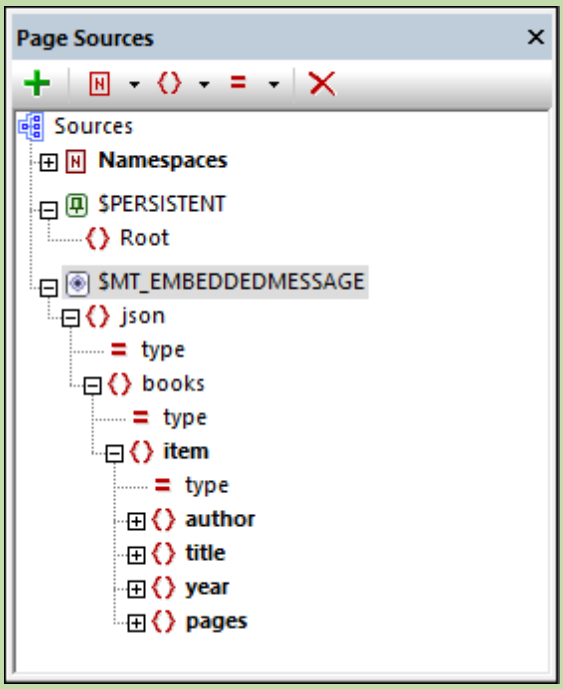
```
function sendbooks() {
   document.querySelector('iframe').contentWindow.postMessage(books, '*');
   }
```

{books} is automatically sent to the workflow on the server (in serialized JSON form).

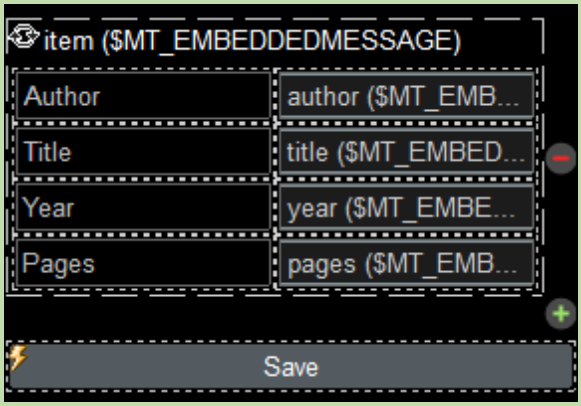
Since page event **OnEmbeddedMessage** has an action defined, ...



... 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.)

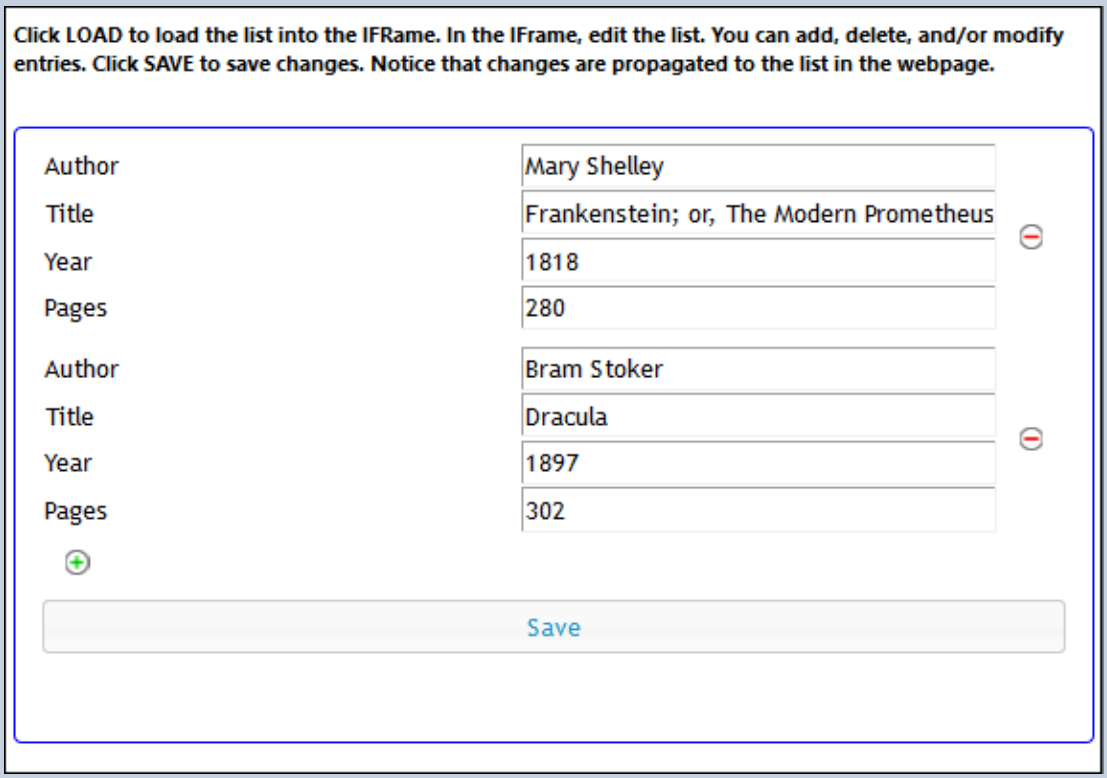


The design contains a repeating table of **item** nodes. Cells of the table are linked, respectively, to the **author**, **title**, **year**, and **pages** page source nodes.



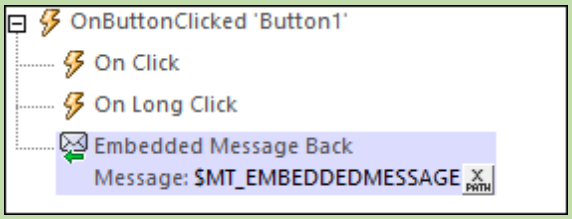
As a result,...

...the solution in the IFrame is updated. The repeating table is populated with the data in the workflow's **$MT_EMBEDDEDMESSAGE** page source.



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.)



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();
}
```

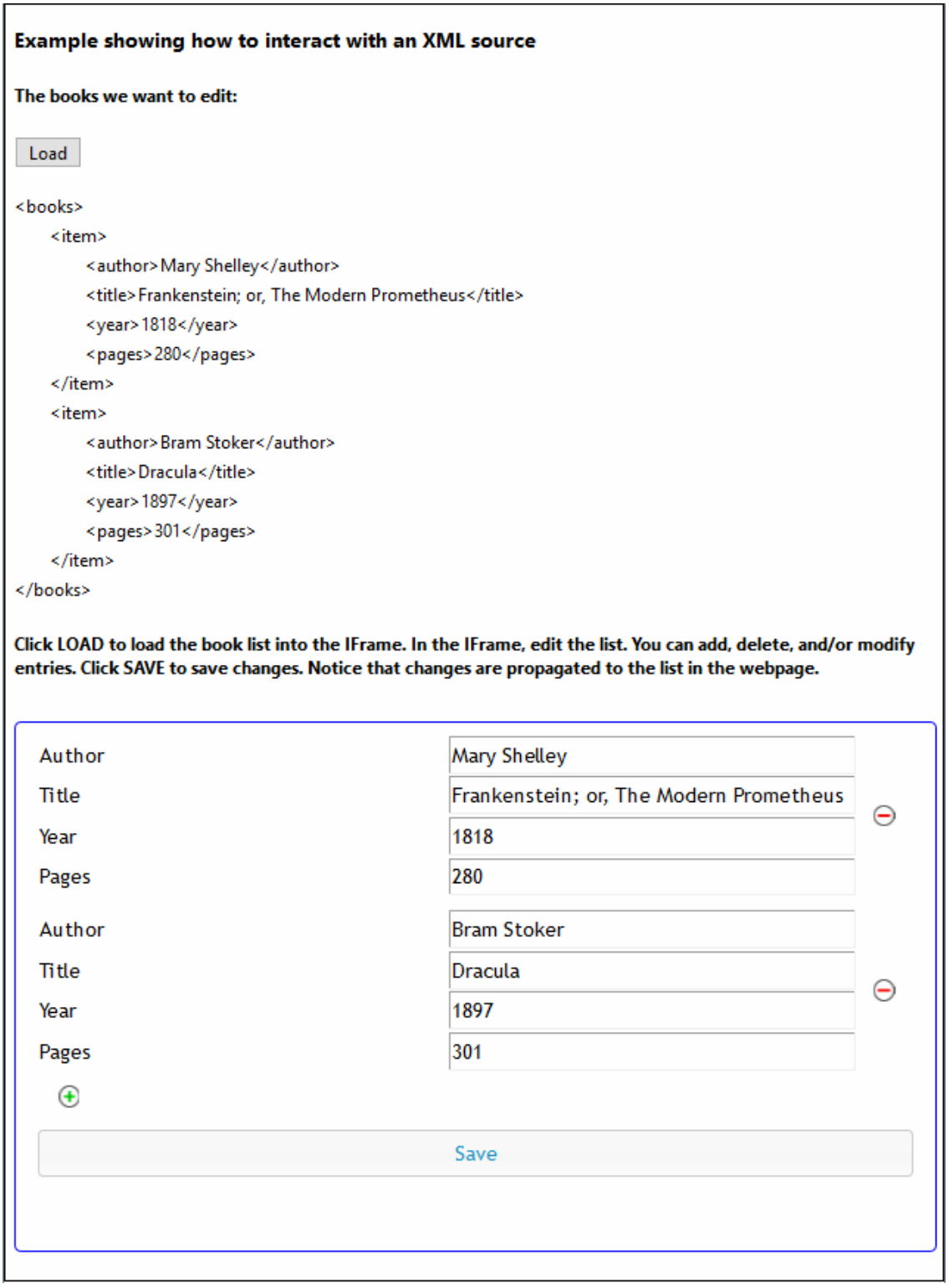
The **showbooks** function displays the updated book list in the webpage:

```
function showbooks() {
   document.querySelector('code').innerText = JSON.stringify(books, null, '
');
}

<pre><code></code></pre>
```

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.



The embedded webpage solution consists of the HTML webpage (**`xmlBooks.html`**) and a MobileTogether design (**`xmlBooks.mtd`**). Both files are located in your *(My) Documents* MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials`**

\EmbeddedWebpageSolutions. To try out the files, deploy the MTD file to your server and enable it to be accessed anonymously. If needed, modify the HTML code so that the IFrame correctly targets the workflow on the server. 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:*

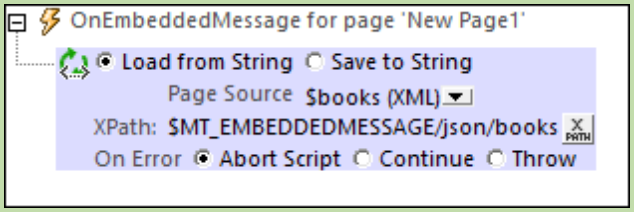
```
<button onclick="sendbooks()">Load</button>
```

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.)

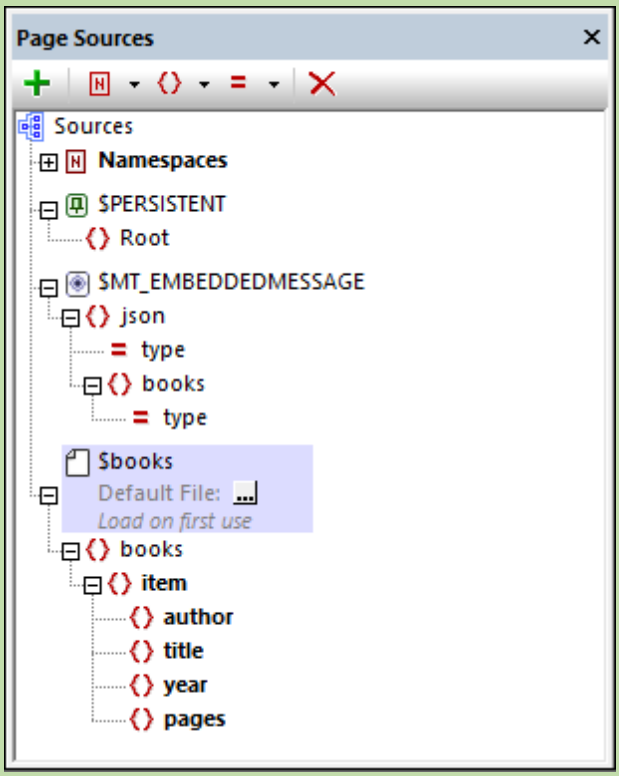
```
function sendbooks() {
   document.querySelector('iframe').contentWindow.postMessage({
      "books": books.childNodes[0].outerHTML
   }, '*');
}
```

{books} is automatically sent to the workflow on the server (in serialized JSON form).

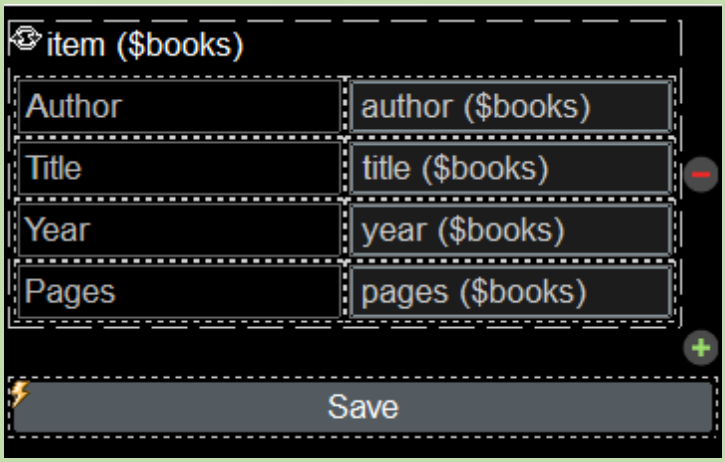
The page event **OnEmbeddedMessage** is enabled since an action has been defined for it (*see screenshot below*). As a result, the page source **$MT_EMBEDDEDMESSAGE** is automatically loaded with **{books}** data.



Th 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*).

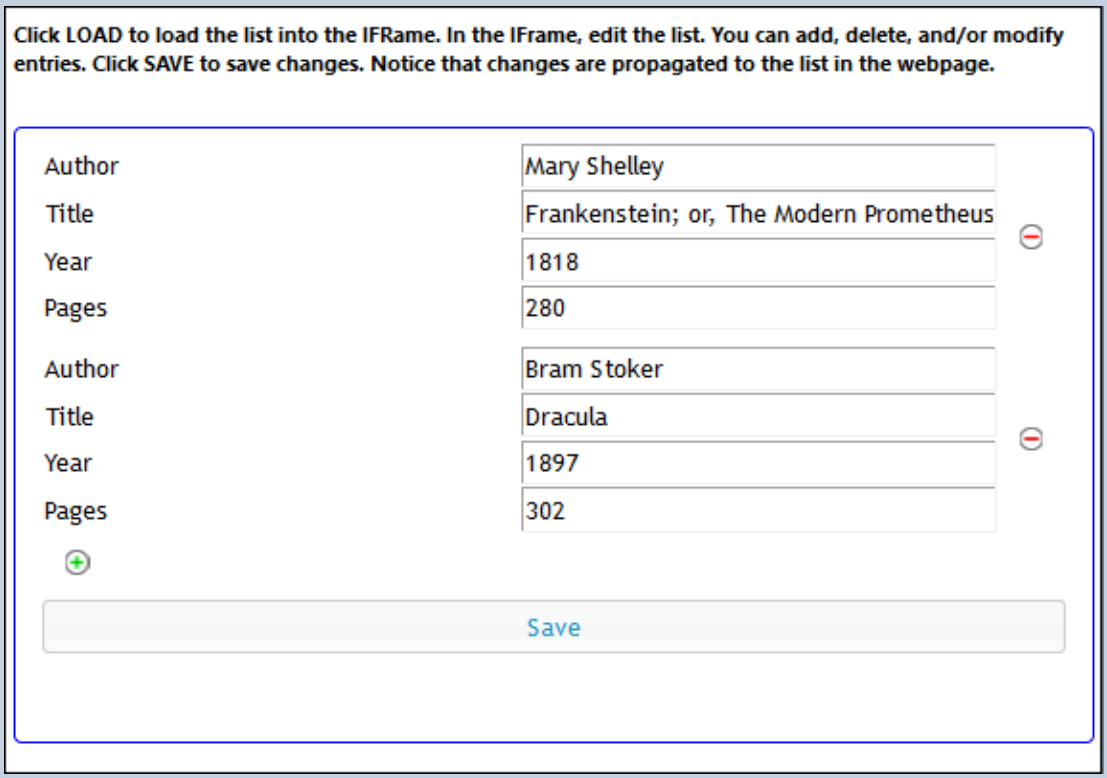


The XML page source has been created so that the XML data can be linked to design components, thereby enabling the XML data to be edited. The design contains a repeating table of **item** nodes of the **$books** page source (*see screenshot below*). Cells of the table are linked, respectively, to the **author**, **title**, **year**, and **pages** page source nodes.



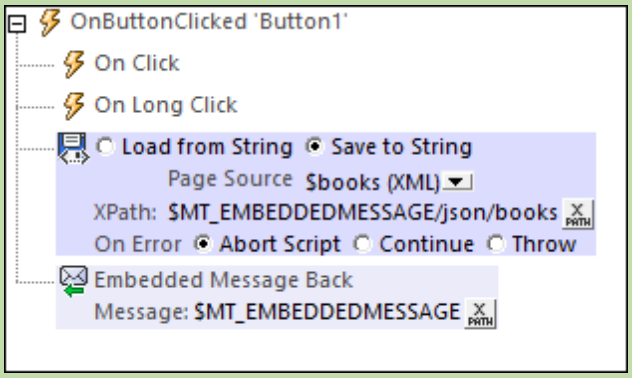
Because the page source has been loaded, the repeating table is populated with the data in the workflow's **$MT_EMBEDDEDMESSAGE** page source.

This data update is shown in the solution in the IFrame.



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.)



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.

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. 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 *(My) Documents* MobileTogether folder: **MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions**. To try out the files, deploy the MTD file to your server and enable it to be accessed anonymously. 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.

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/js4.0/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.

```
<script src="http://localhost:8083/js4.0/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);
      }
```

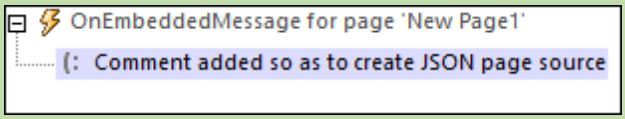
```
      _data = data;
      _jwt = jwt;
   if (_jwt) {
         url += '&auth';
      }
         iframe.addEventListener('load', _start);
            iframe.src = url + '&embed';
   }

   this.post = function(data) {
      iframe.contentWindow.postMessage(data, '*');
   }
}
```

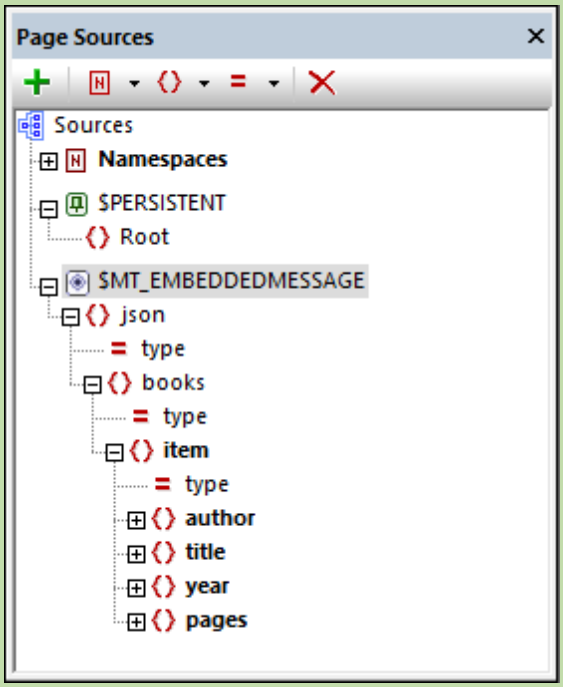
The IFrame loads the solution **jsonBooks** and receives the data from the webpage.

{books} is automatically sent to the workflow on the server (in serialized JSON form).

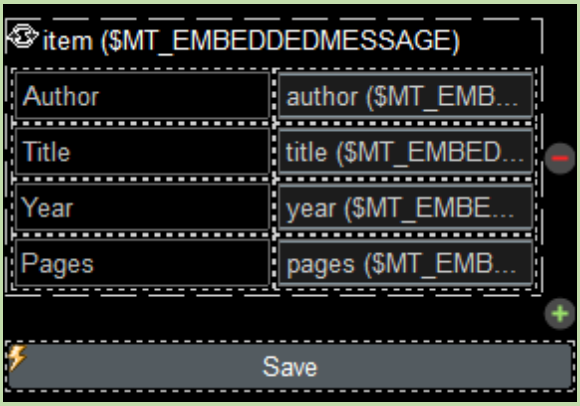
Since page event **OnEmbeddedMessage** has an action defined, ...



... the page source **$MT_EMBEDDEDMESSAGE** is automatically loaded with **{books}** data.

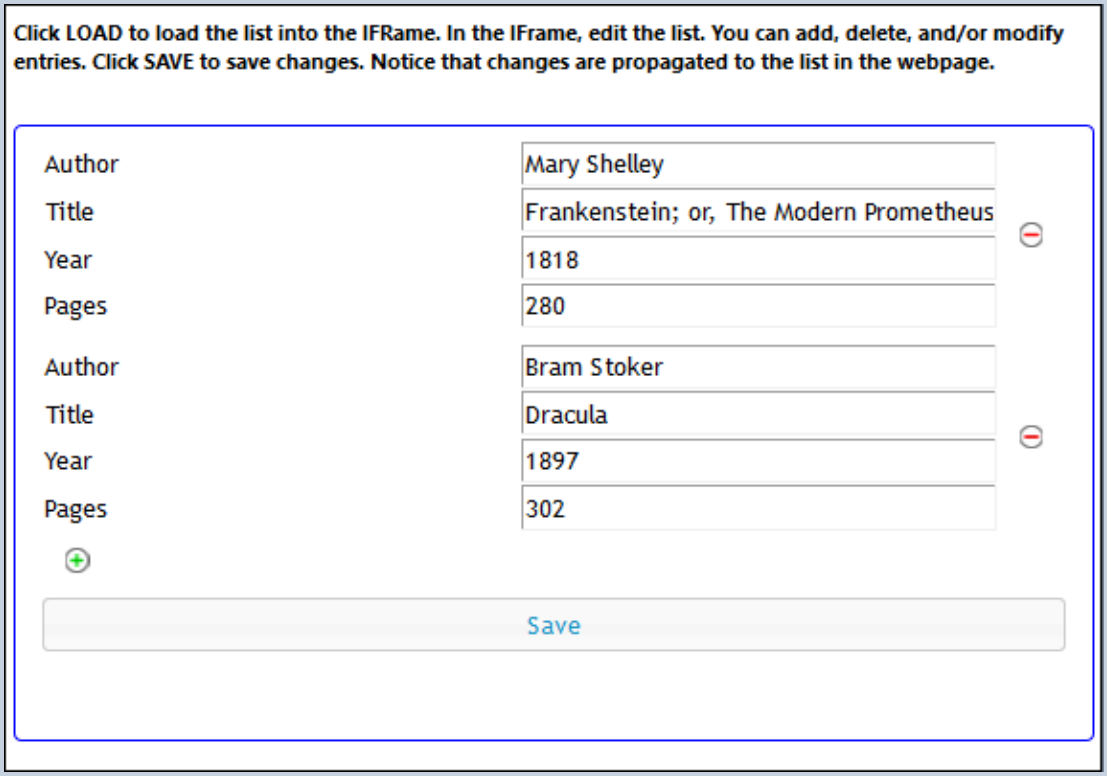


The design contains a repeating table of **item** nodes. Cells of the table are linked, respectively, to the **author**, **title**, **year**, and **pages** page source nodes.



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.



The data has been made available in the IFrame directly on opening the HTML page.

JWT Authentication

The JWT authentication example in this section modifies the webpage of the Pre-setting page source example in the previous section. 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.

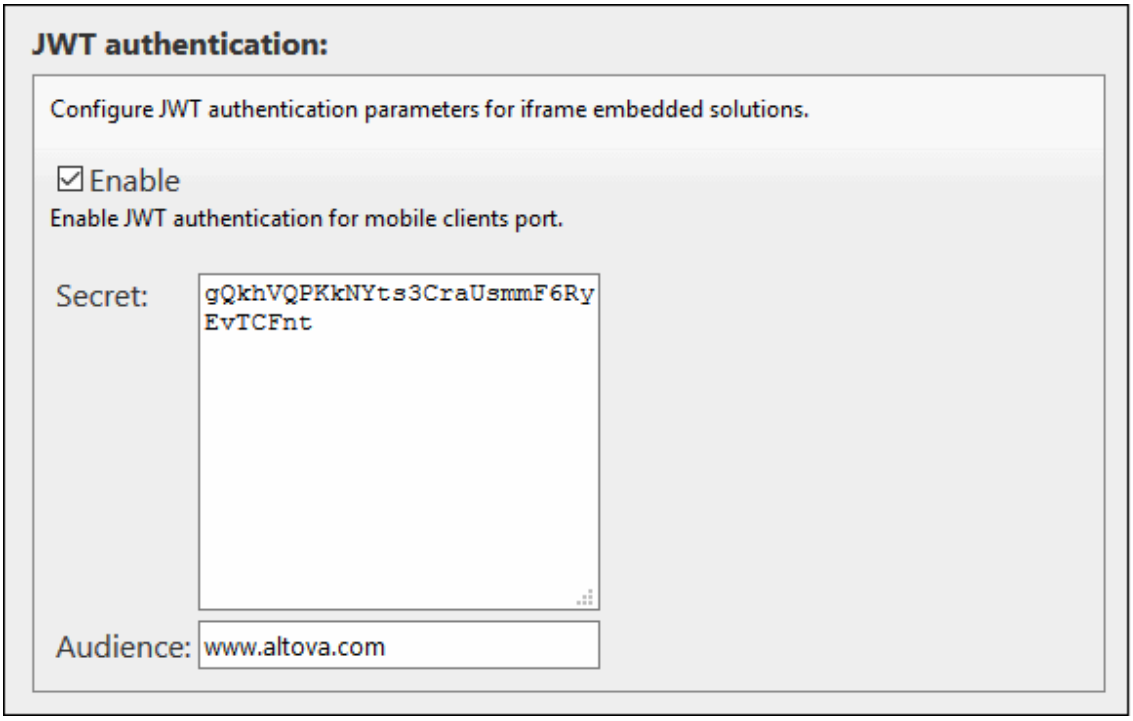
The files used in this example are **`JWT.html`** and **`jsonBooks.mtd`**. Both are located in your *(My) Documents* MobileTogether folder: **`MobileTogetherDesignerExamples\Tutorials\EmbeddedWebpageSolutions`**. To try out the files, deploy the MTD file to your server, and enable JWT authentication 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`**



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/js4.0/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.eyJpc3MiOiJPbmxpbmUgSldUIEJ1aWxkZXIiLCJp
YXQiOjE1MDE4MzU4MzUsImV4cCI6MTUzMzM3MTgzNSwiYXVkIjoid3d3LmFsdG92YS5jb20iLCJzdW
IiOiJuZXd1c2VyIiwiR2l2ZW5OYW1lIjoiSm9obm55IiwiU3VybmFtZSI6IlJvY2tldCIsIkVtYWls
IjoianJvY2tldEBleGFtcGxlLmNvbSIsIlJvbGUiOlsiTWFuYWdlciIsIlByb2plY3QgQWRtaW5pc3
RyYXRvciJdfQ.M66SBrP_U30iQeheQWpPTdaBzsJZqK2L7BJQ8gKP-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>
```

Chapter 17

Automated Testing

17 Automated Testing

The Automated Testing feature enables you to compare two test runs (which are essentially simulations) 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. 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. 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 **Project** menu. They are also available via the Automated Testing toolbar (*screenshot below*).

	*Record New Test Case:* Starts a new test case in the Simulator 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 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.*
	*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.*
	*Manage Test Cases and Runs:* Displays the *Manage Test Cases and Runs* 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
- Playing Back a Test Case
- Managing Test Cases and Runs
- Deploying Test Cases to Server
- Comparing Test Runs

17.1 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 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 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. 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 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 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 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.

17.2 Playing Back a Test Case

After a test case has been recorded (*see Recording a Test Case*), 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 as a **test run** that is based on its originating **test case** (notice the terminology; *see Managing Test Cases and Runs* 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. If the playback returns differences, then the playback is saved.

Playback options

Options for recording and playback are available at the bottom of the Manage Test Cases and Runs dialog. 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 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 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 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. If the playback returns differences, then the playback is automatically saved as a test run when it is closed.

Playing back a test case on a client device

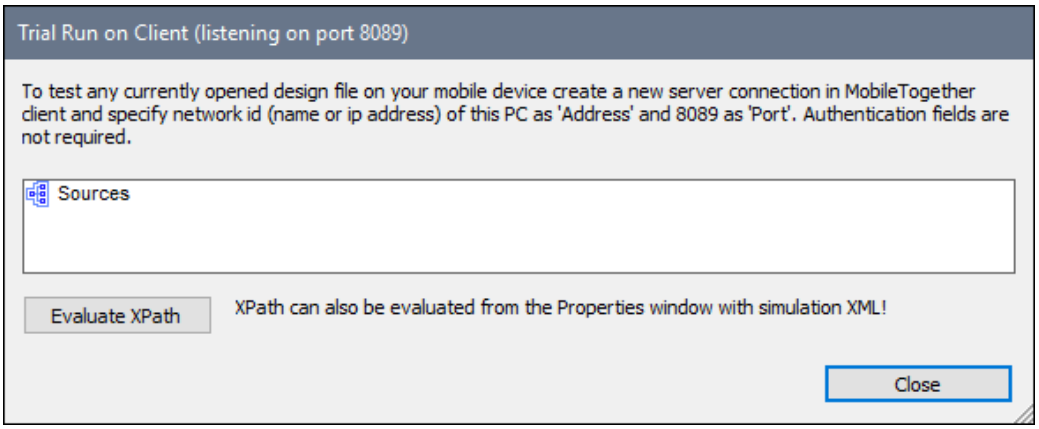
Playing back a test case on a client enables you to preview and check whether a given test case functions correctly on specific client devices. In this scenario, the MobileTogether Designer machine takes on the role of a MobileTogether Server. This means that you must set a separate port via which the 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**, and set up a separate port for communication with the client device: for example, **`8089`**.
- 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*).



3. On the client device, connect to the Designer machine and refresh the solutions. 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.

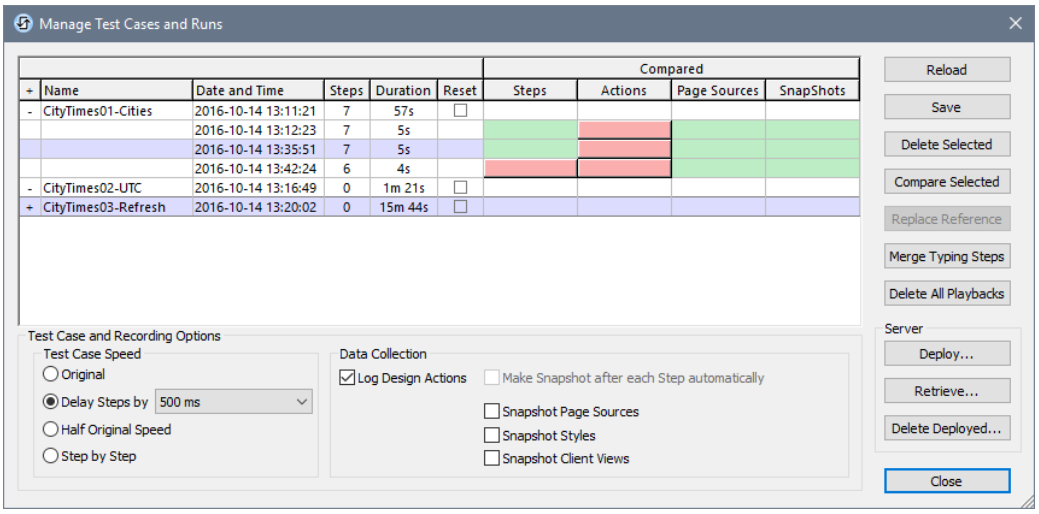
The test case will be played back on the client. In MobileTogether Designer, the Trial Run on Client dialog (*screenshot above*) will be updated with page source data as the playback progresses. If the playback returns differences, then the playback is automatically saved (as a test run) when it is closed. To check whether there were differences, click **Manage Recordings** in the *Automated Testing* toolbar.

17.3 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.



The dialog displays each recorded test case and its associated test runs (playbacks), 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. 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 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 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 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. 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 next section 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 their check boxes and click **Delete Checked**. 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 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.

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*).

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**.

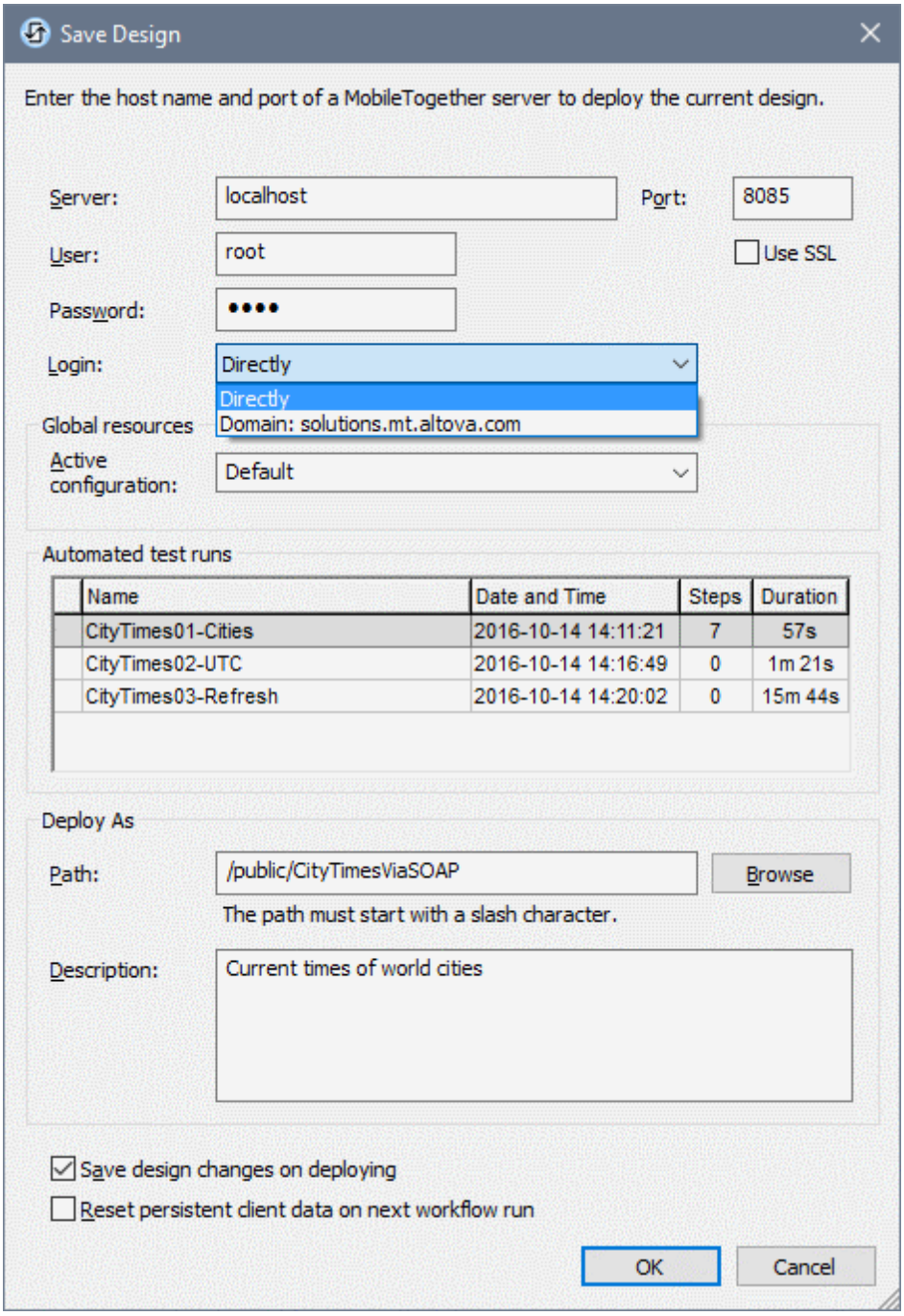
For more details about these procedures, see Deploying to Server.

17.4 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

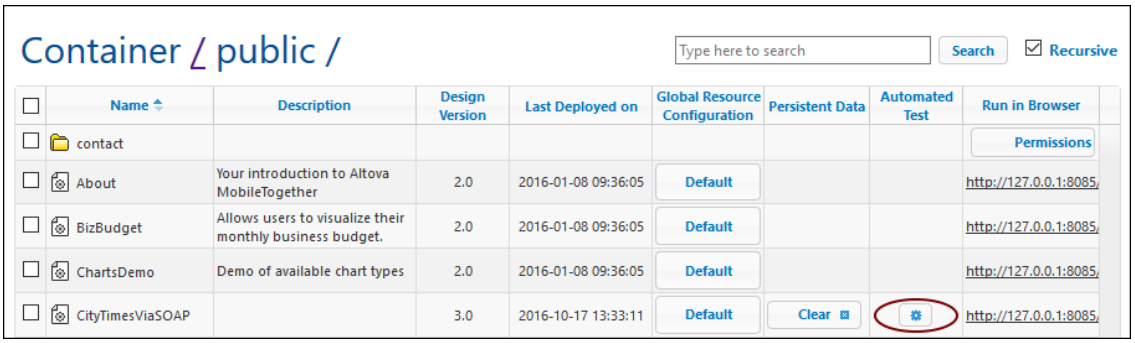
A test case can be deployed with a design to MobileTogether Server via the **File | Deploy to MobileTogether Server** command. This command displays a dialog (*screenshot below*) in which the following are entered: (i) the server's access details, (ii) the path to the design on the server, and (iii) the test cases to deploy with that design (select the test cases that you want to deploy).



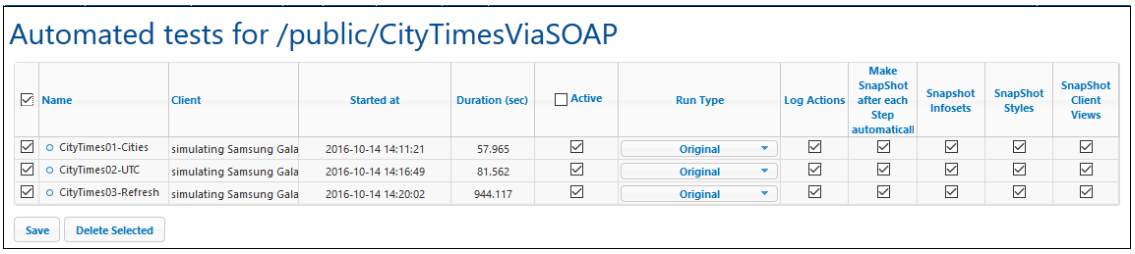
If a design has already been deployed to the server, you can additionally deploy its test cases by clicking **Deploy** in the Manage Test Cases and Runs dialog. Test cases are identified by internal IDs, so only test cases with the same ID are overwritten.

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*).



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*).



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:

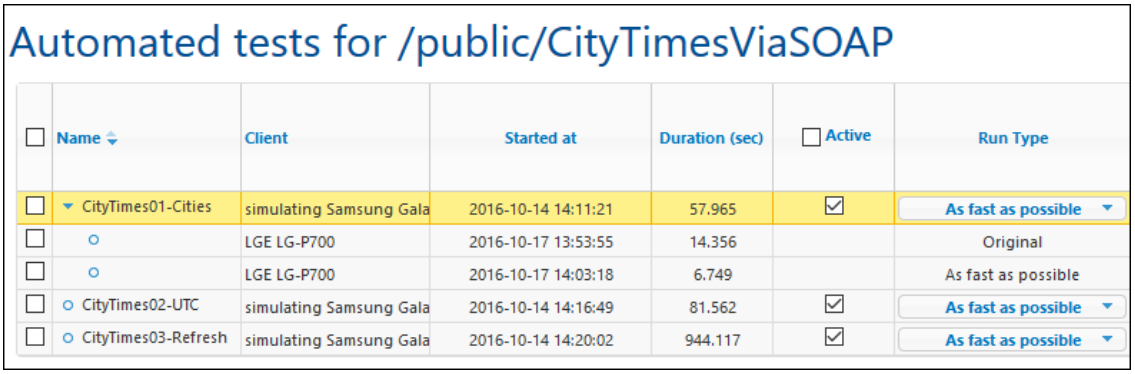
1. In the *Active* column, check the test cases that you want to make active. These test cases will be played back on the client when the user starts a solution. If multiple test cases are selected, then all the selected test cases will be played back. If any one of a solution's test cases has been activated, 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 for information about these options.
3. Click **Save** to finish.

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 (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

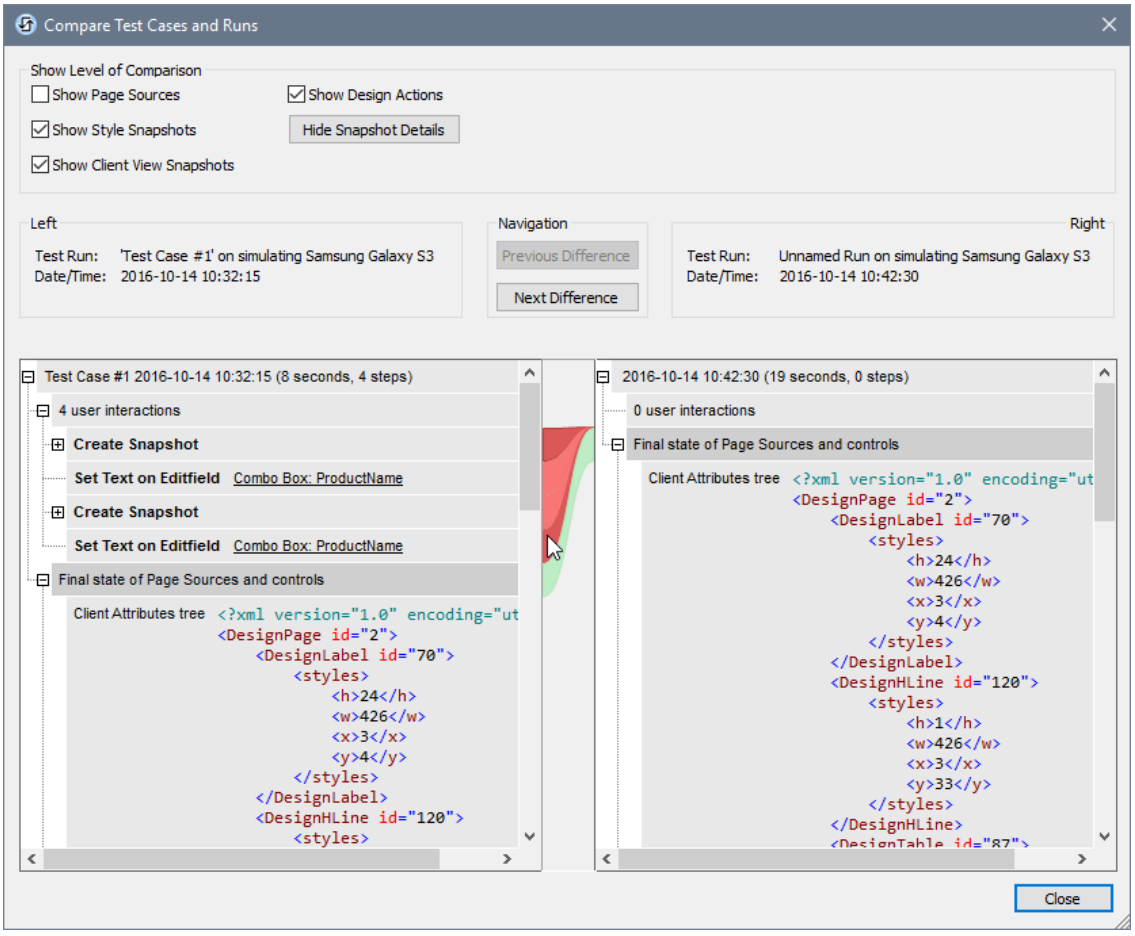
test run and displayed on the Automated Tests page (*screenshot below*). Each playback result is shown as a descendant of its test case. For example, in the screenshot below, it can be seen that two test runs have been stored for the test case `CityTimes01-Cities`. Notice also that the client device 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**.



You can retrieve test runs to MobileTogether Designer by clicking **Retrieve** in the Manage Test Cases and Runs dialog (of MobileTogether Designer). Retrieval applies to all test runs of all the deployed test runs 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.

17.5 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, 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.

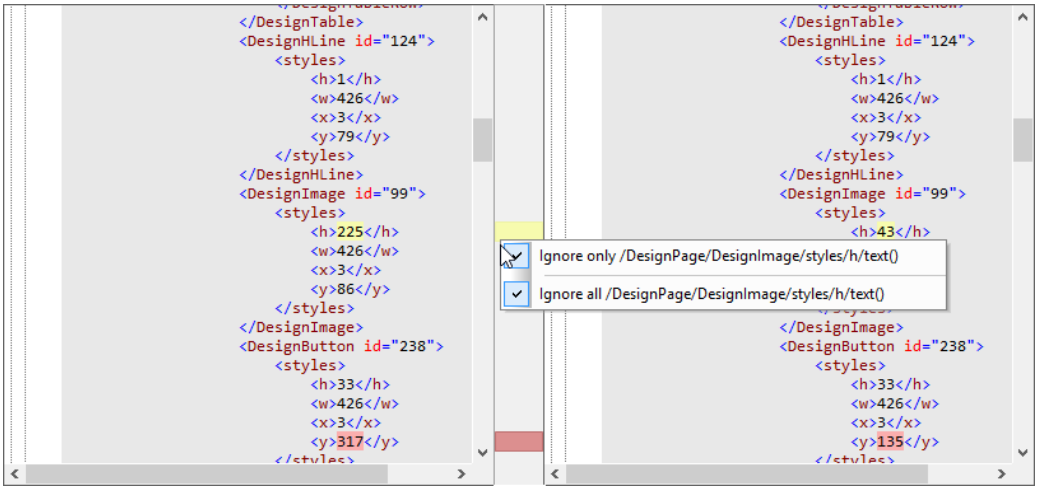


Each test run is displayed in a separate pane as a tree (*see screenshot above*), with their details displayed above the pane. Within the pane, each tree is structured as a chronological sequence of user actions (steps) and design actions. If a snapshot was taken at some point during the test run (for 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 dark red connector between the panes (*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*>**.



- 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.

Chapter 18

AppStore Apps

18 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 of your appstore app from the design via the **File | Generate Program Code for AppStore Apps** command. Program code can be generated for Android, iOS, Windows Phone, and Windows.
3. Deploy the project to MobileTogether Server. This is done in the same way as for other project deployments.
4. Compile the generated program code to build your appstore app for the respective devices and operating systems.

The program-code generation and code-compilation 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.

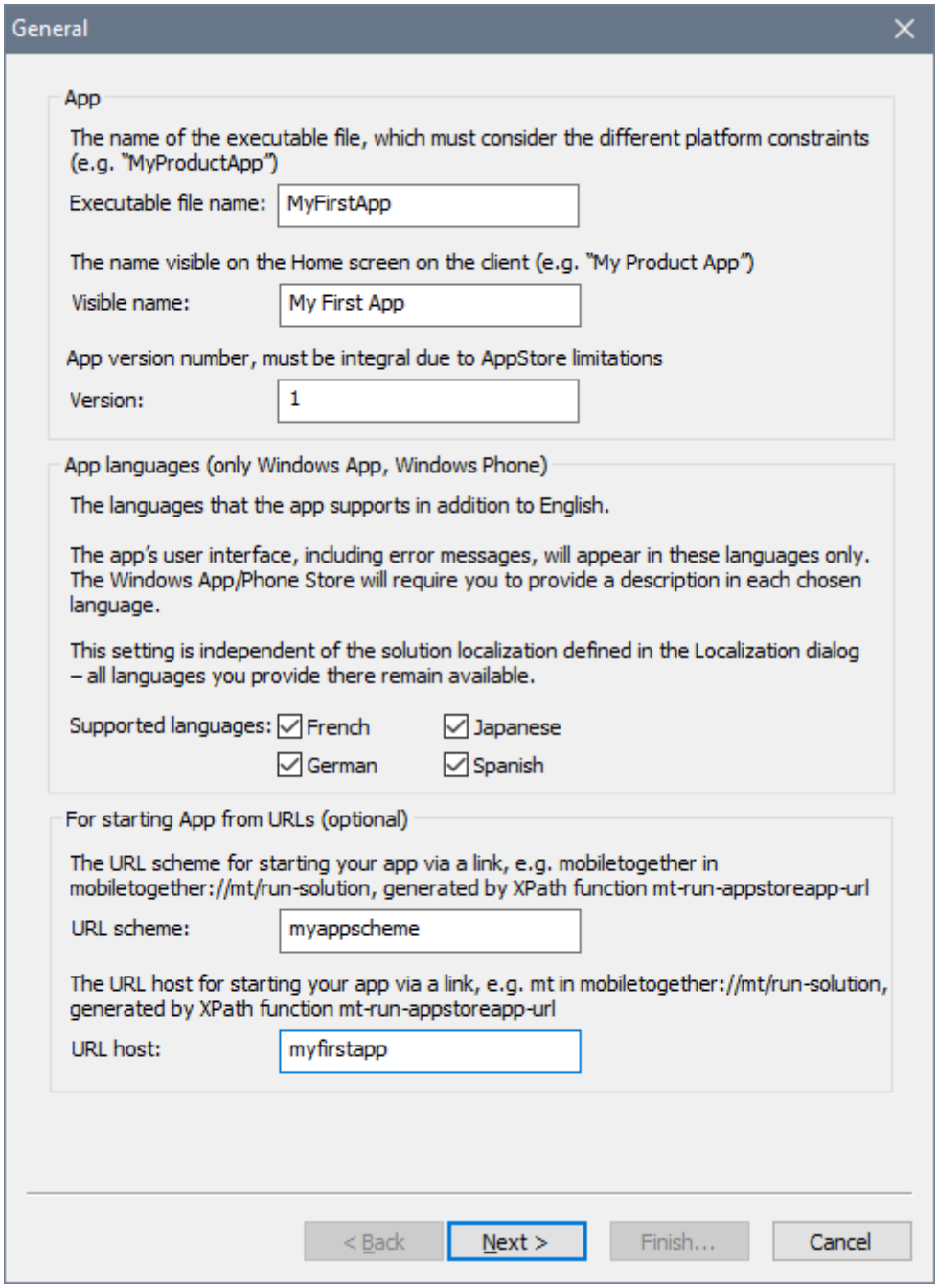
The differences between the solution and the appstore app are laid out in the table below.

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

18.1 Generate Program Code from Project

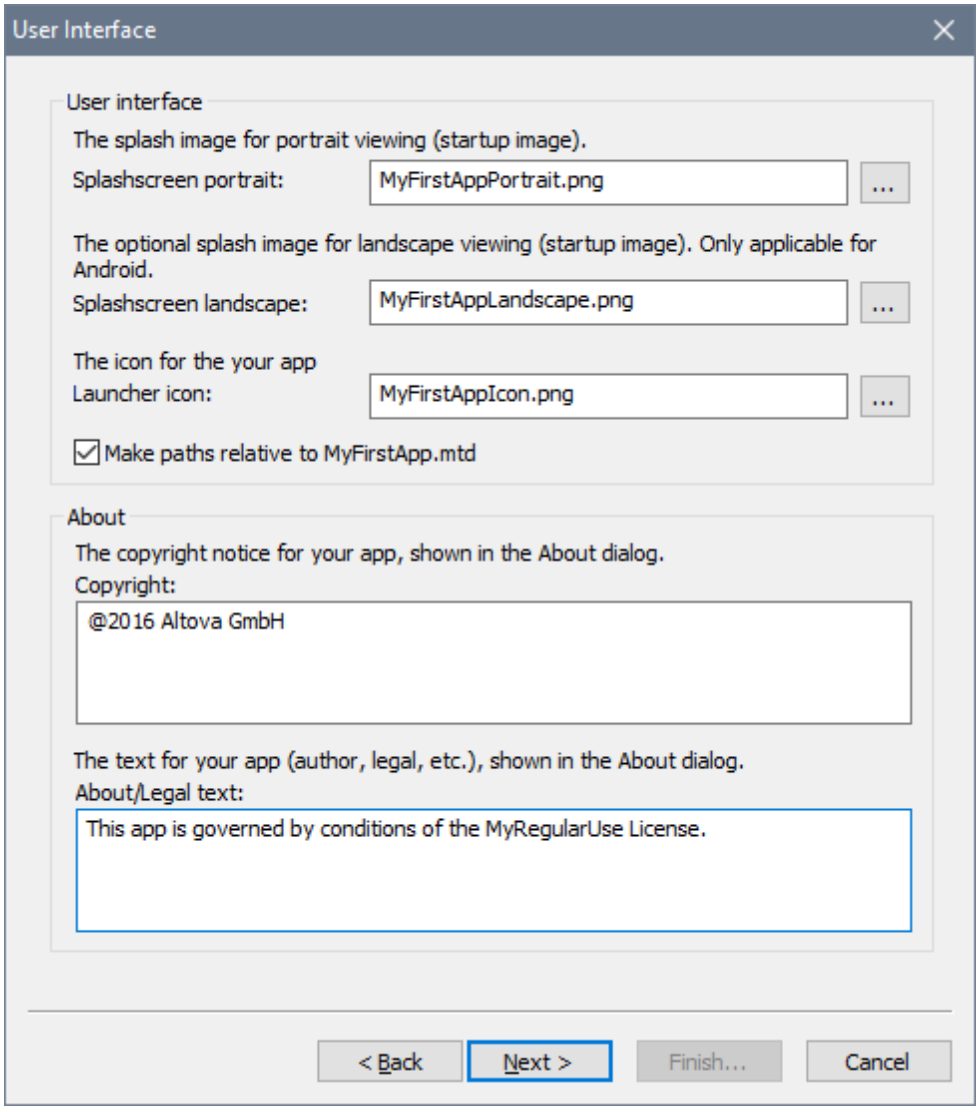
To generate program code for appstore apps that run on Android, iOS, Windows Phone, and Windows mobile devices, click the command **File | Generate Program Code for AppStore Apps**. The command opens a wizard that has seven screens if all mobile formats are selected. Each screen contains settings for the generated code.

▼ 1: General: Names, Version, Languages, URL



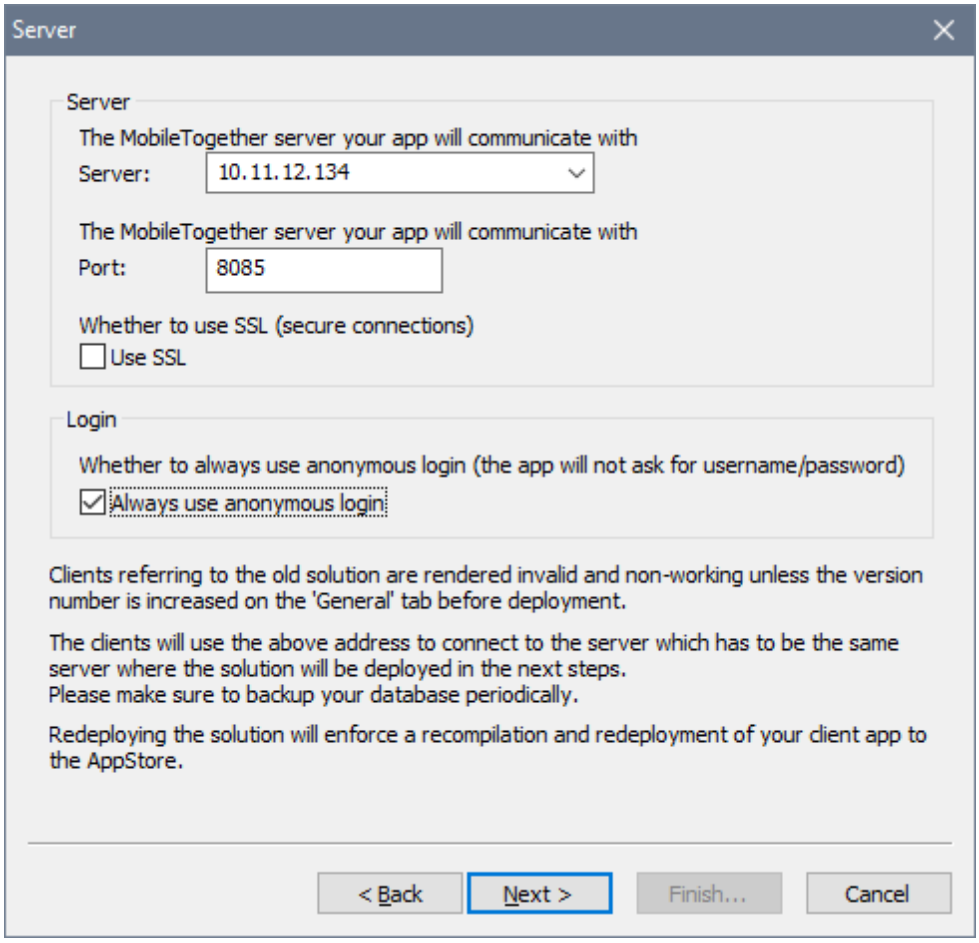
- *Executable file name:* The name that is used internally to reference the code. You should use a name that has no spaces.
- *Visible name:* The name of the app that will be visible to the user.
- *Version:* The version number of the app. Must be an integer or a decimal number. For example: `1` or `1.0` 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.
- *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



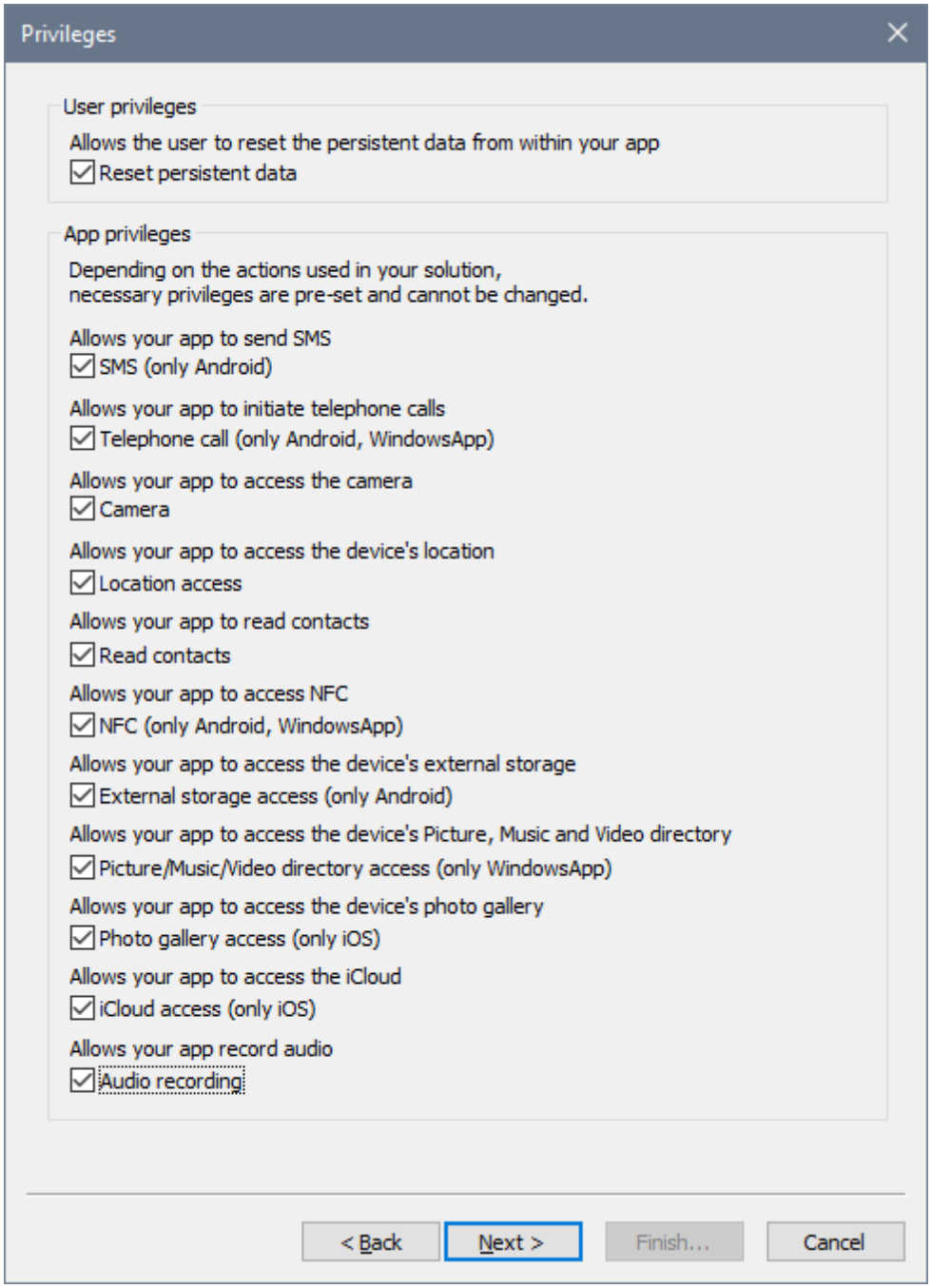
- *Splashscreens:* Browse for the splashscreens that should be used in portrait and landscape orientations. The corresponding splashscreen will be used on the mobile device when the device's orientation changes. For iOS apps, the portrait splashscreen is drawn in both orientations to aspect-fill the 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.
- *Launcher icon:* The icon that is displayed in the apps screen of the mobile device to launch the app. The maximum pixel size is `200x200`.
- *Copyright and legal notice:* The text to be displayed in the mobile device.

▼ 3: Server: Server and Login Settings



- *Server:* The IP address of the server on which the workflow will be deployed.
- *Port:* The port on the server 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.
- *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.

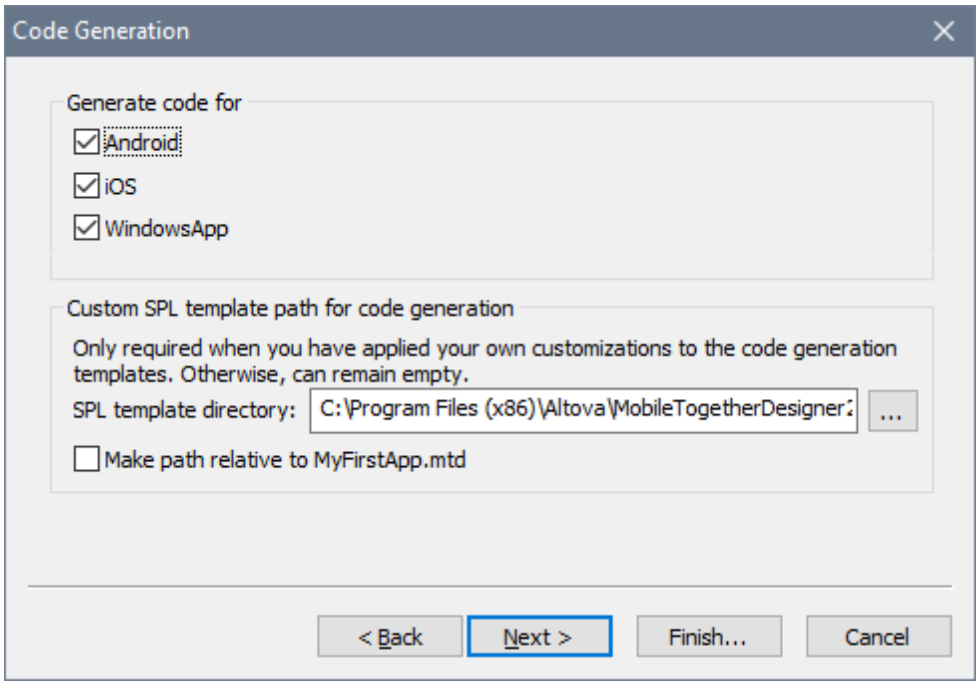
▼ 4: Privileges: User and App privileges



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* 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

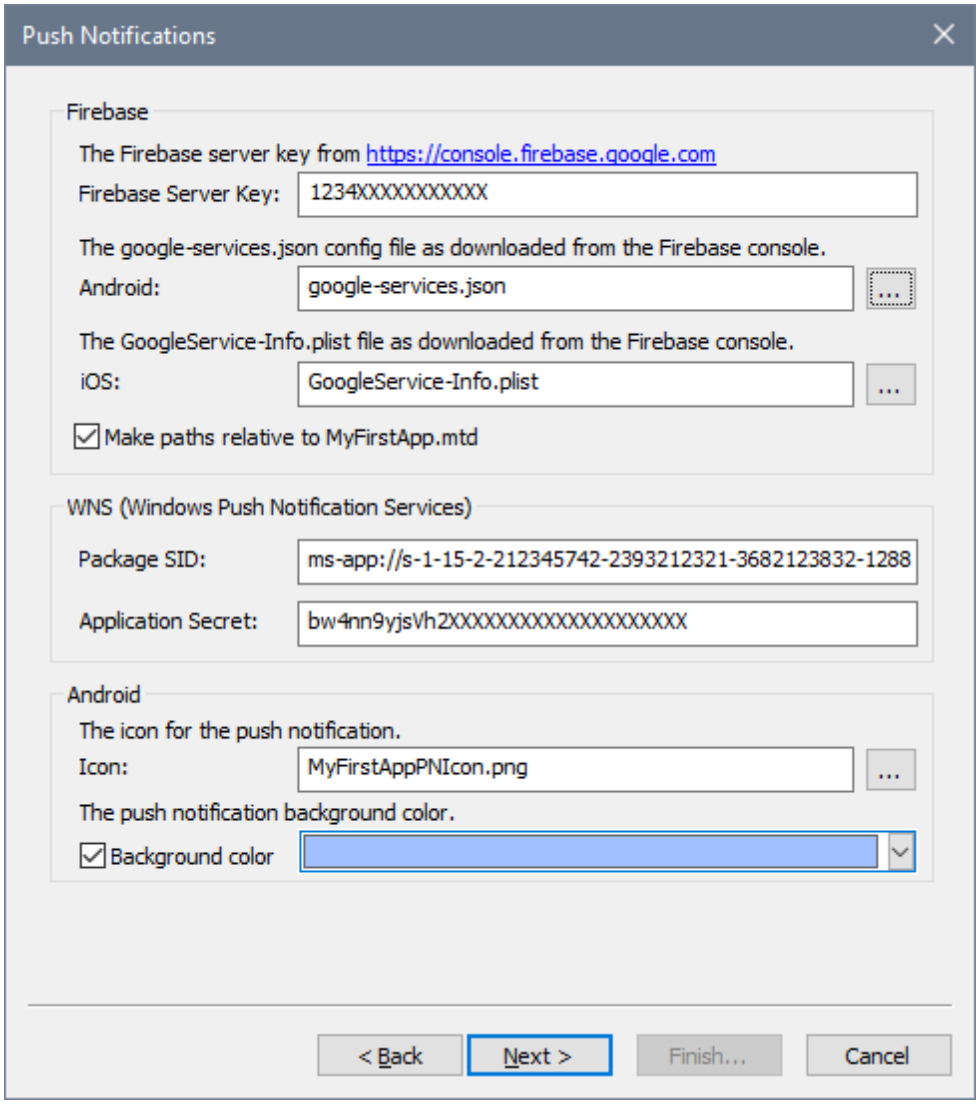
geolocation features, and so requires access to the geolocation information of the device. Because of this the *Location access* privilege is automatically selected, and you cannot change it.

▼ 5: Code Generation: App Formats and SPL Template Location



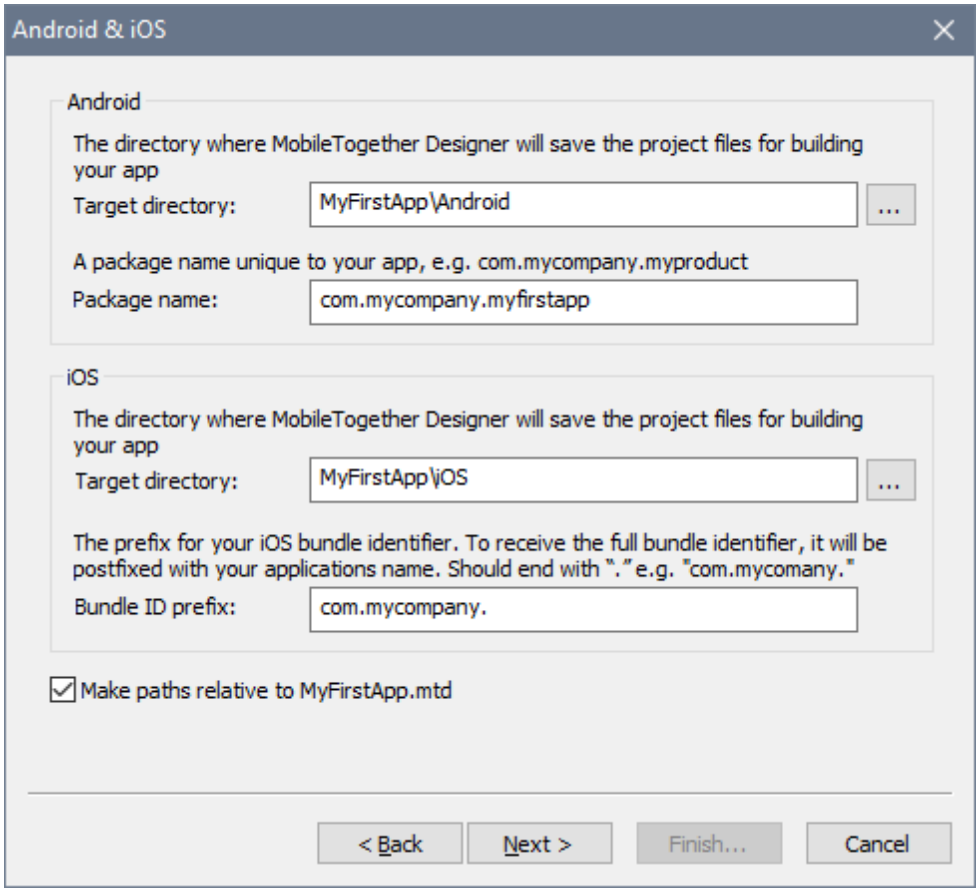
Select the app formats for which you want to generate program code. SPL templates for the different app formats are required for generation of the program code. These are provided with your MobileTogether Designer installation and are located here: `C:\Program Files (x86)\Altova\MobileTogetherDesigner4\MobileTogetherSPL`. If you customize one or more SPL templates, create a copy of the SPL template directory (in which the customized files will be saved) and specify the location of this directory in this screen (*see screenshot above*). The new folder must reproduce the directory structure of the original SPL template directory.

▼ 6: Push Notifications



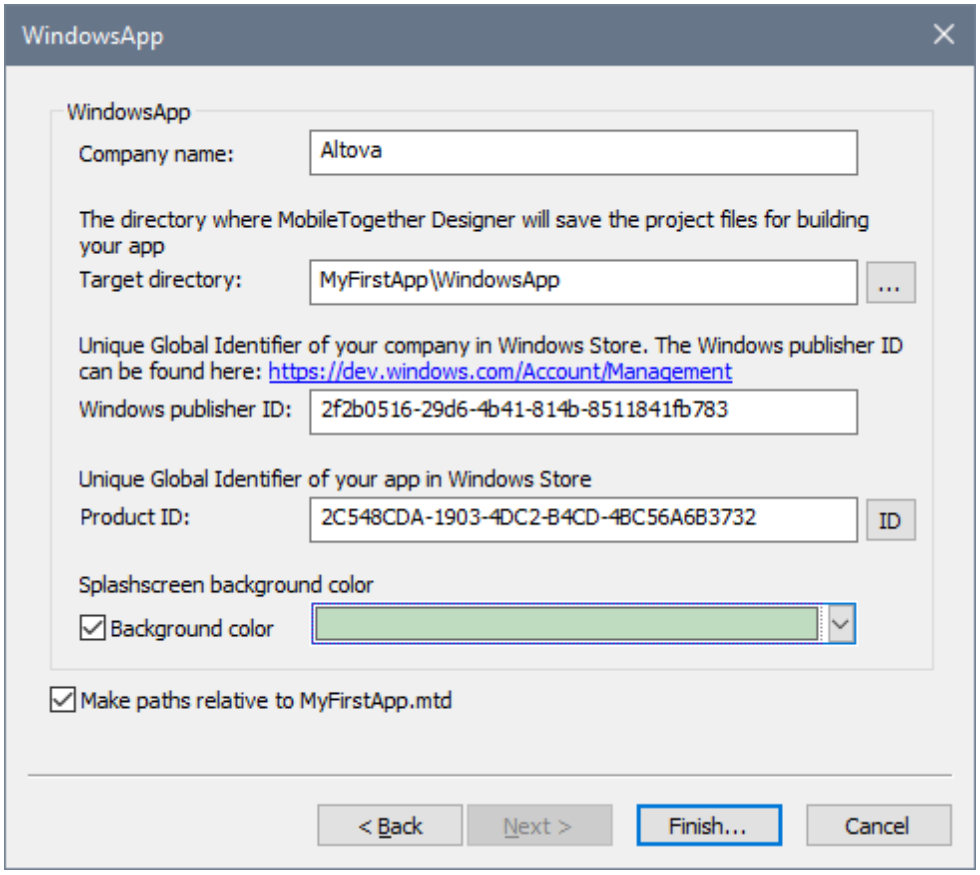
This screen appears only if the **`OnPushNotificationReceived`** event has an action defined for it. In this screen, enter the details you obtained when making the push notification (PN) registrations for the different operating systems (*see Push Notifications in AppStore Apps*). For PNs that are sent to Android devices, you can also select a background color for the PN.

▼ 7: Android and iOS



Sets the target directory where the program code will be generated for the respective formats. You must specify the package name for the Android package and 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 automatically created in the target directory of the program code. *See Compile Program Code: iOS for more information.*

▼ 8: WindowsApp

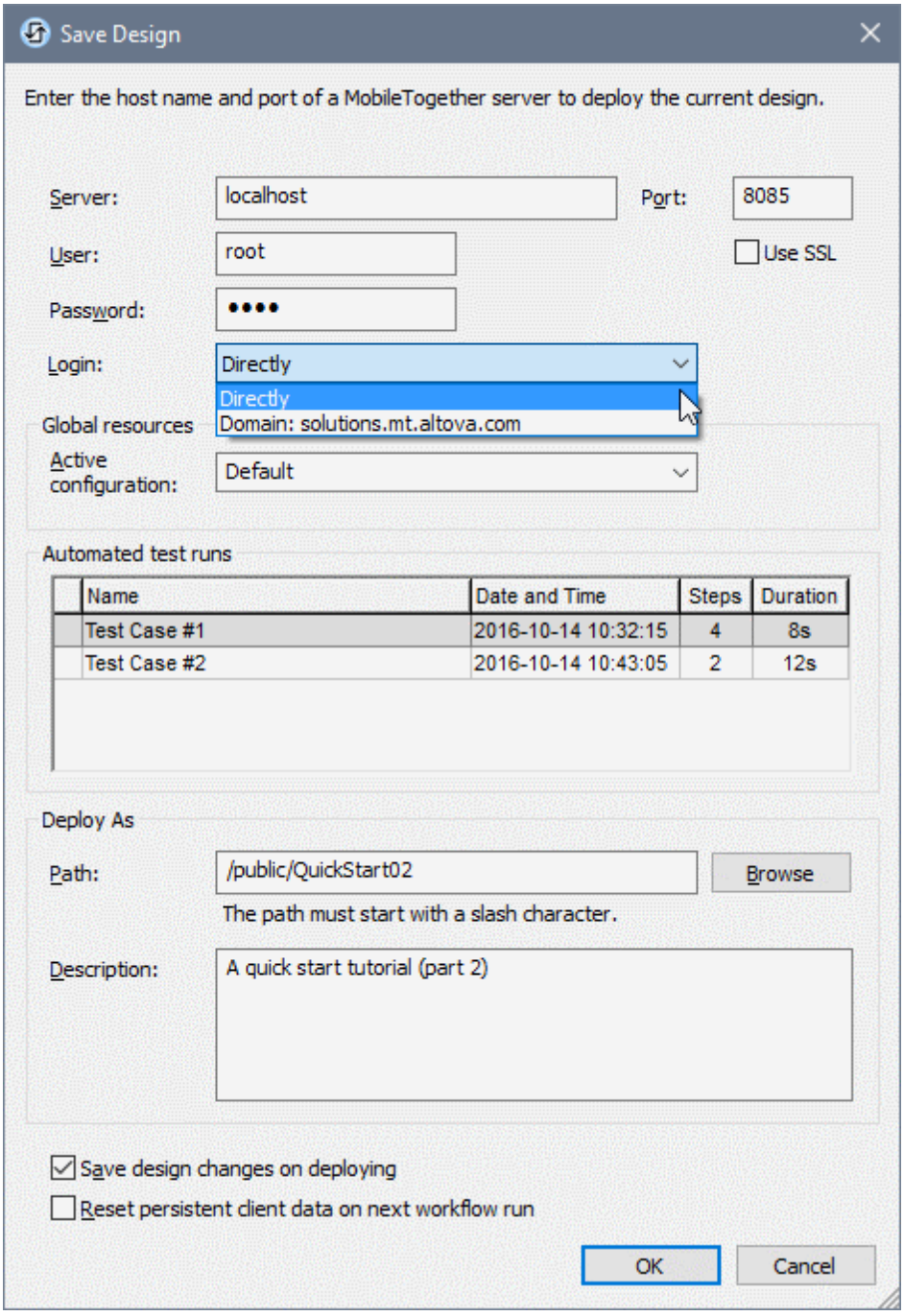


Sets the target directory where the program code will be generated for the Windows App format. You must specify your company's Windows Publisher ID and a product ID. The Publisher ID is the GUID that's assigned to your developer account (*see Windows App | Requirements*; the Windows Publisher ID can be found on your Account Summary page in the Dev Center; the link in the dialog takes you there). The Product ID is used for identifying the app. You can also select a background color for the splashscreen of the app.

After you complete entering the information required by the Generate Program Code Wizard, click **Finish**. The wizard closes, and the Deploy App to Server dialog appears.

18.2 Deploy Workflow to Server

When you click **Finish** in the Generate Program Code Wizard, the Deploy App to MobileTogether Server dialog (*screenshot below*) appears. Fill in the data fields of the dialog as described in the description of the **Deploy to MobileTogether Server** 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 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 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) 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.

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.

18.3 Compile Program Code

After you complete the Generate Program Code Wizard and click **OK** in the Deploy Workflow to Server dialog, 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. 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. SPL and SPL templates are described in the section SPL Templates.

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
- iOS
- Windows App

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).

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 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.

app\src\main\java\<package-name>\MainActivity.java
app\src\main\java\<package-name>\<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.

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.

Building the Android app

The app will be built with Android Studio. The steps described here apply to Android Studio 1.3 RC 4, which is the version that is current at the time of writing (October 2015).

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: **app\build\outputs\apk\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\app-release.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.

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. 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 9.0 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

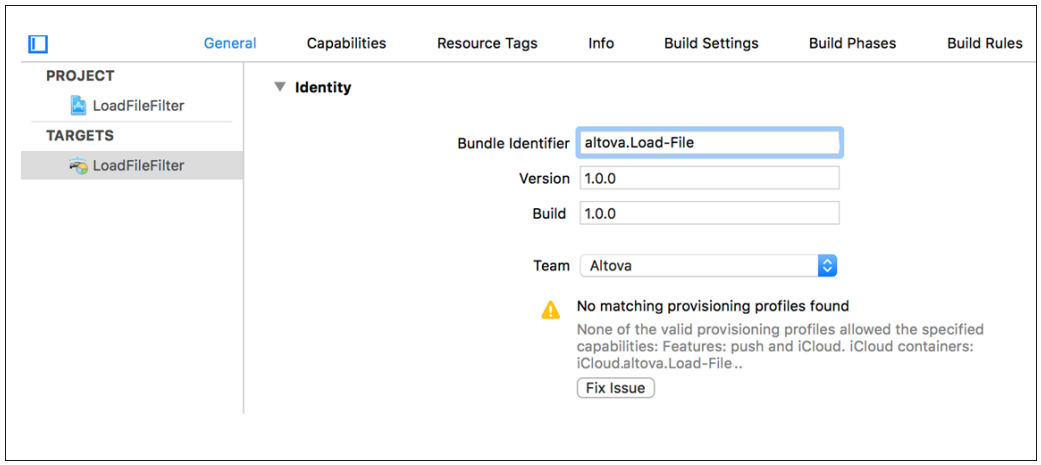
1. Open the `.xcodeproj` file in XCode.
2. Ensure that team, app id, and provisioning profiles are set up. See Configuring Your XCode Project for Distribution for information about how to do this. Note that test devices for ad hoc distribution are only active after re-generating the corresponding provisioning files.
3. Finalize what splashscreen/s to use. By default, the portrait splashscreen (specified in Screen 2 of the Generate Program Code Wizard) 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*. Note that it is currently not possible to run the compiled app in the simulator. So if you want to test the app, you must connect an 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.



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. 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.

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. 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.

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 or Visual Studio 2017. For each version the respective additional components listed below are also required.
  - *Visual Studio 2015 Update 3:* (i) Universal Windows App Development Tools (1.4.1) and Windows 10 SDK (10.0.14393); (ii) Windows 8.1 and Windows Phone 8.0/8.1 Tools: Tools and Windows SDKs
  - *Visual Studio 2017:* (i) C++ Universal Windows Platform Tools; (ii) Windows 10 SDK (10.0.14393.0); (iii) Windows 8.1 SDK (*can be found under "SDKs, libraries, and frameworks" in "Individual components"*)
- Windows 8.1 or higher for testing the app.

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.
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. 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.

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. If the former, then edit the SPL template/s. Run the Generate Program Code Wizard 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.

18.4 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, Windows, and Windows Phone.

- The SPL templates are delivered with your installation of MobileTogether Designer and are located in the folder **C:\Program Files (x86)\Altova\MobileTogetherDesigner4\MobileTogetherSPL**.
- The SPL templates use variables that are tied to information that you enter in the Generate Program Code Wizard.

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\MobileTogetherDesigner4\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 can 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 with the modified SPL templates. In Screen 5 of the Generate Program Code Wizard, 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:

- SPL Syntax
- SPL String Mechanisms
- Properties of $Options
- Properties of $Application
- Miscellaneous Objects

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:

`[$x = 42` `$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*.
- Object: Provided by MobileTogether. For example, the `$Options` 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]
```

String Mechanisms

SPL provides the string manipulation mechanisms that are listed below.

---

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 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 the string contained in deploymentPath as XML-escaped text.

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, 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:* `$Options.workflowKey` returns the workflow key. Every time program code is generated and the associated workflow is deployed to the server, both are assigned the same unique workflow key. An appstore app will be able to access this workflow only if it has the same key as the workflow. See Deploy Workflow to Server for details.
- **deploymentPath**: returns the path of the deployed workflow. *Example:* `$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.

appName
visibleAppName
appVersion
hostForRunSolutionUrl
schemeForRunSolutionUrl

User interface properties

The values of these properties are provided in Screen 2 of the Generate Program Code Wizard.

splashScreenPortraitFilePath
splashScreenLandscapeFilePath

launcherIconFilePath
aboutLegal
aboutCopyRight

Server properties

The values of these properties are provided in Screen 3 of the Generate Program Code Wizard.

serverAddress
serverPort
isServerAccessAlwaysAnonymous
isUseSSL

Properties about user and app privileges

The values of these properties are provided in Screen 4 of the Generate Program Code Wizard.

mayResetPersistentData
isAllowSMS
isAllowTelephoneCall
isAllowCamera
isAllowLocationAccess
isAllowExternalStorageAccess
isAllowiCloudAccess
isAllowAudioRecording
isAllowPhotoGalleryAccess

Android and iOS properties

The values of these properties are provided in Screen 6 of the Generate Program Code Wizard.

targetDirectoryAndroid
androidPackageName
androidPackageDir
targetDirectoryIOS
iosBundleIdentifierPrefix

Windows App and Windows Phone properties

The values of these properties are provided in Screen 7 of the Generate Program Code Wizard.

windowsAppCompanyName
windowsPhoneCompanyName
windowsCompanyPublisherID
targetDirectoryWindowsApp

windowsAppCompanyProductID
targetDirectoryWindowsPhone
windowsPhoneCompanyProductID

Properties of $Application

The **$Application** object can take the properties listed below. The object's properties can be accessed by using the **.** operator. Some SPL template usage examples are given below.

```
sub CopyFile( byval $sSource, byval $sTarget )
    return $Application.CopyFile( $sSource, $sTarget )
endsub

sub UnzipFile( byval $sZipFile, byval $sTargetDir )
    return $Application.UnzipFile( $sZipFile, $sTargetDir )
endsub
```

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`.

DeleteFile(sFile)
Deletes the file at the location `sFile`.

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`.

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

Chapter 19

Server Services

19 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 conditions are defined in the administrator interface of MobileTogether Server Advanced Edition.

The broad steps for creating and running a MobileTogether service is as follows:

1. Create a service in MobileTogether Designer. Each service is stored in a single `.mtd` file.
2. Deploy the service 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.

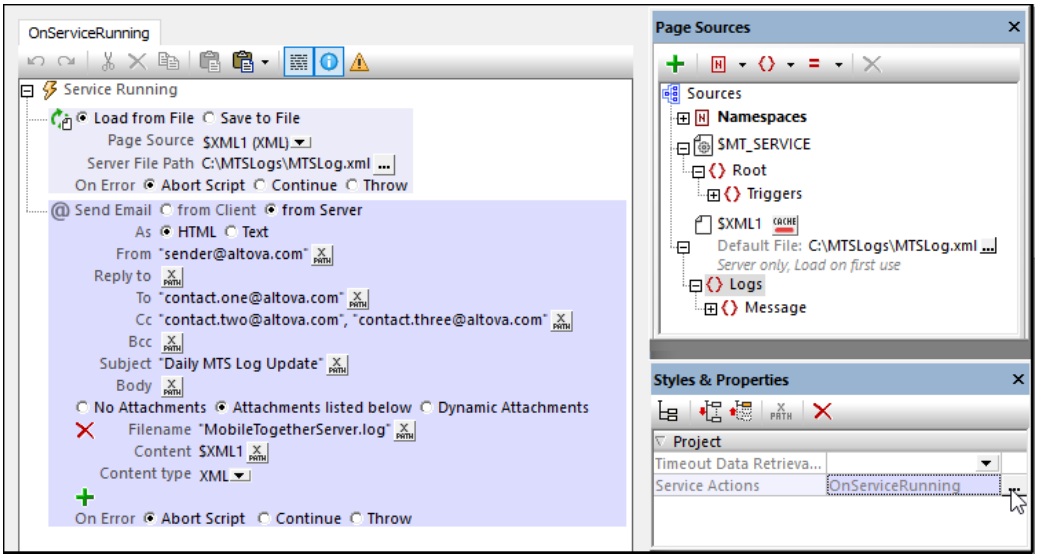
Each of these steps is described in more detail in the sub-sections of this section. For information about setting up a service on MobileTogether Server, see the MobileTogether Server documentation.

19.1 Creating a Service

A service is created in the same way as you create a solution. You can define page source trees 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 and the addition of new pages (whether top or sub) 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** to open a design file for the service. A new **service design** is created, and a **$MT_SERVICE** page source is automatically created. The MobileTogether Designer interface will look the same as that for a solution. One difference you might quickly notice is that no client-interface design is possible since all controls are disabled. Instead, all the actions you want to define for the service must be defined in the tab of the project event `OnServiceRunning`.
2. If you need to use page sources, add them to the Page Sources Pane. In the screenshot below, an XML page source named **$XML1** has been added.
3. Open the Actions dialog of the service (or Service Action Tree, *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, click the **Additional Dialog** button of the **Service Actions** property.



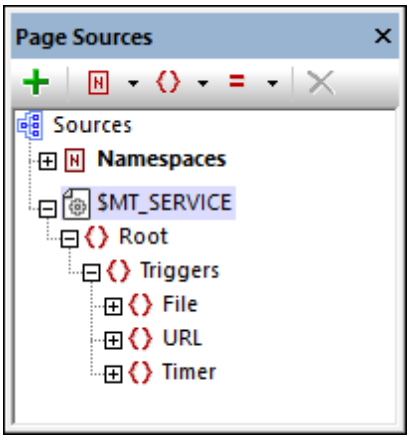
4. The left-hand pane of the dialog (*not shown in screenshot above*) displays all the available actions for services. The unavailable actions are disabled (and shown grayed out). Drag the actions you want to execute as the service into the `OnServiceRunning` tab. In the screenshot above, two actions 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**

a second time, a new empty service file is created.

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 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 building of 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.

Service properties

In the Styles & Properties Pane (*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.

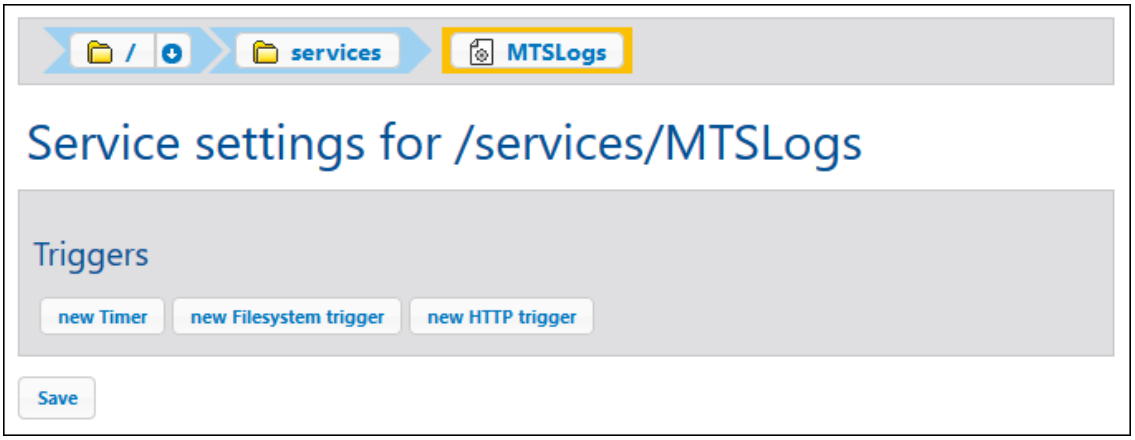
19.2 Deploying a Service

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, by using the **File | Deploy to MobileTogether Server**.

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.

19.3 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.

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.

Chapter 20

Menu Commands

20 Menu Commands

MobileTogether Designer menu commands are organized into the following menus:

- File
- Edit
- Project
- Page
- Table
- View
- Tools
- Window
- Help

20.1 File

The **File** menu contains the following commands:

- New
- New Service
- Open
- Reload
- Close
- Close All
- Close All But Active
- Save
- Save As
- Save Copy As
- Save All
- Deploy to MobileTogether Server
- Open from MobileTogether Server
- Delete from MobileTogether Server
- Generate Program Code for AppStore Apps
- Send by Mail
- Print
- Print Preview
- Print Setup
- Recent Files
- Exit

New

- 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.

New Service

- 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.

Open

- Open
  - *Icon*

  - *Shortcut*

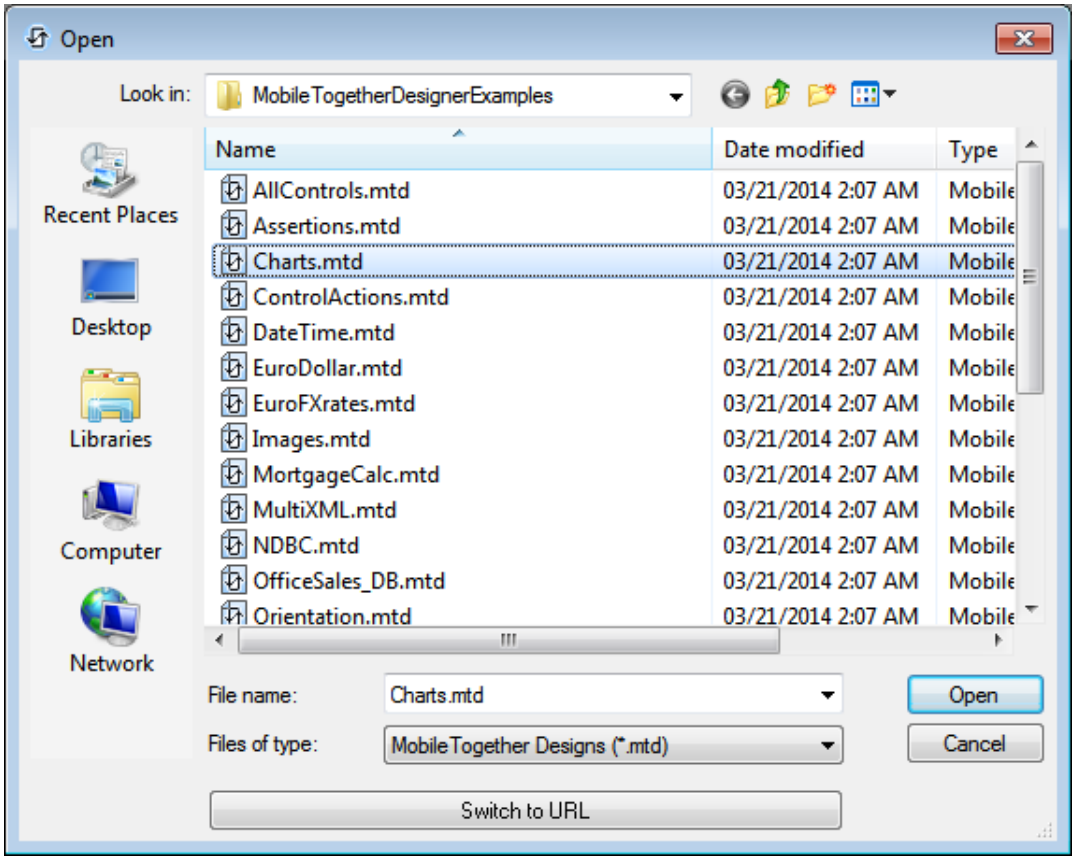
    Ctrl+O

  - *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.

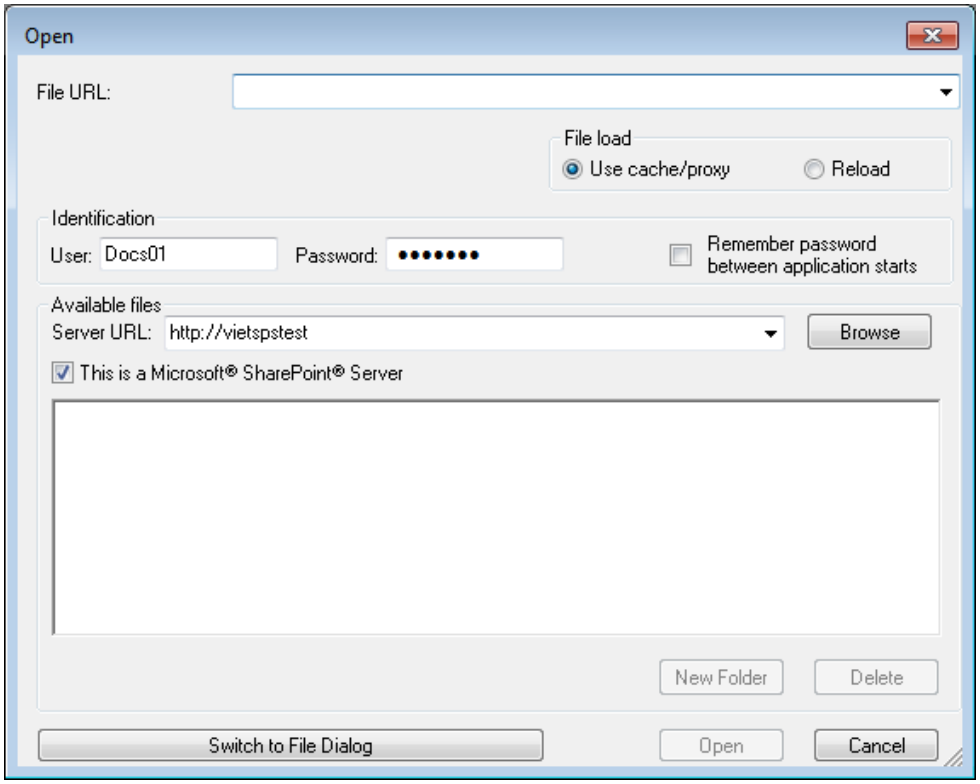
- Selecting and saving files via URLs

  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.



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*).



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.

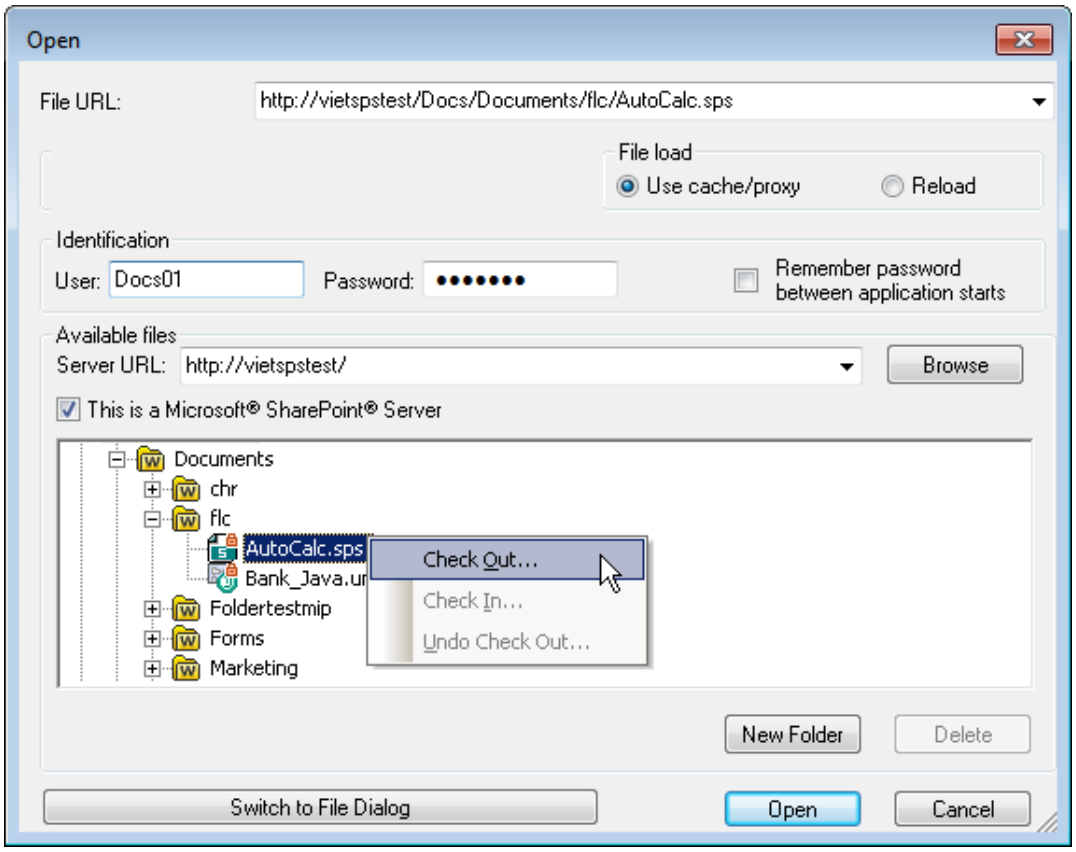
Note the following:

- 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.



Right-clicking a file pops up a context menu containing commands available for that file (*screenshot above*).

- The various file icons are shown below:

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	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**.

Reload

- 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.

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.

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*

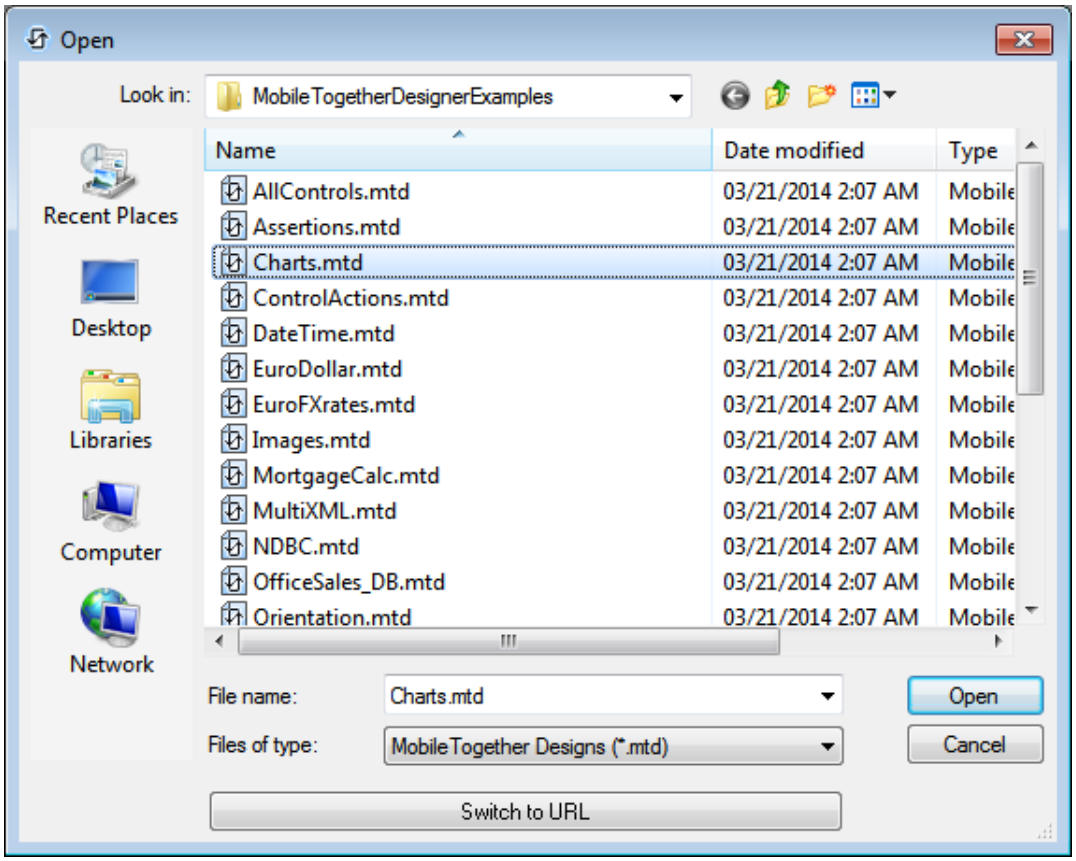
    

  - *Description*

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.

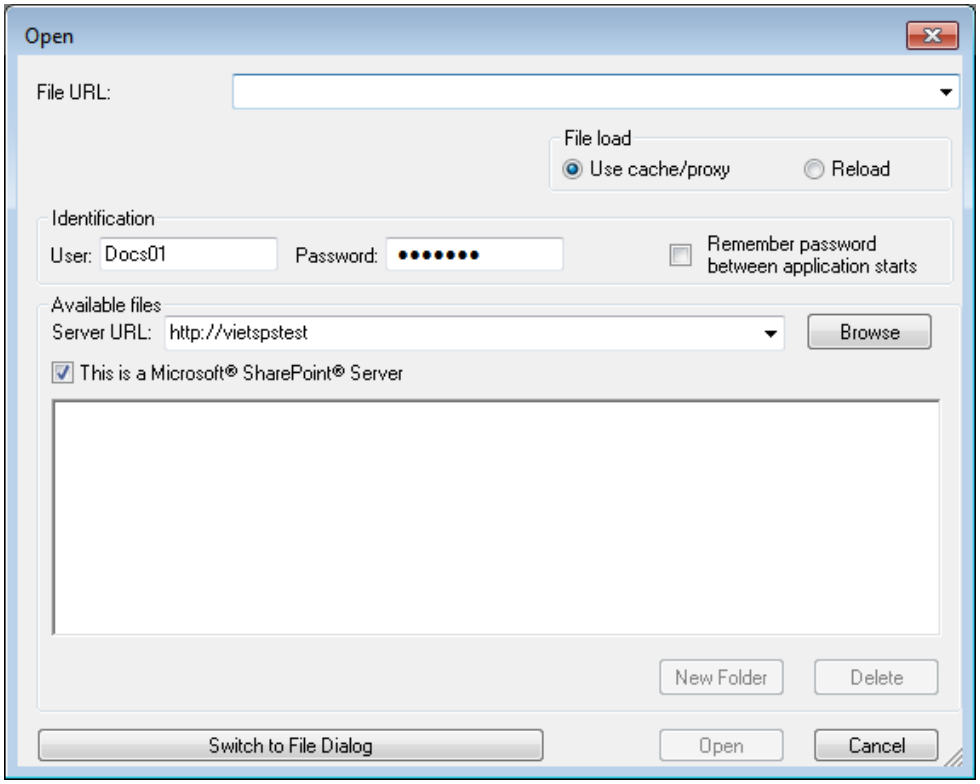
▼ Selecting and saving files via URLs

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.



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*).



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.

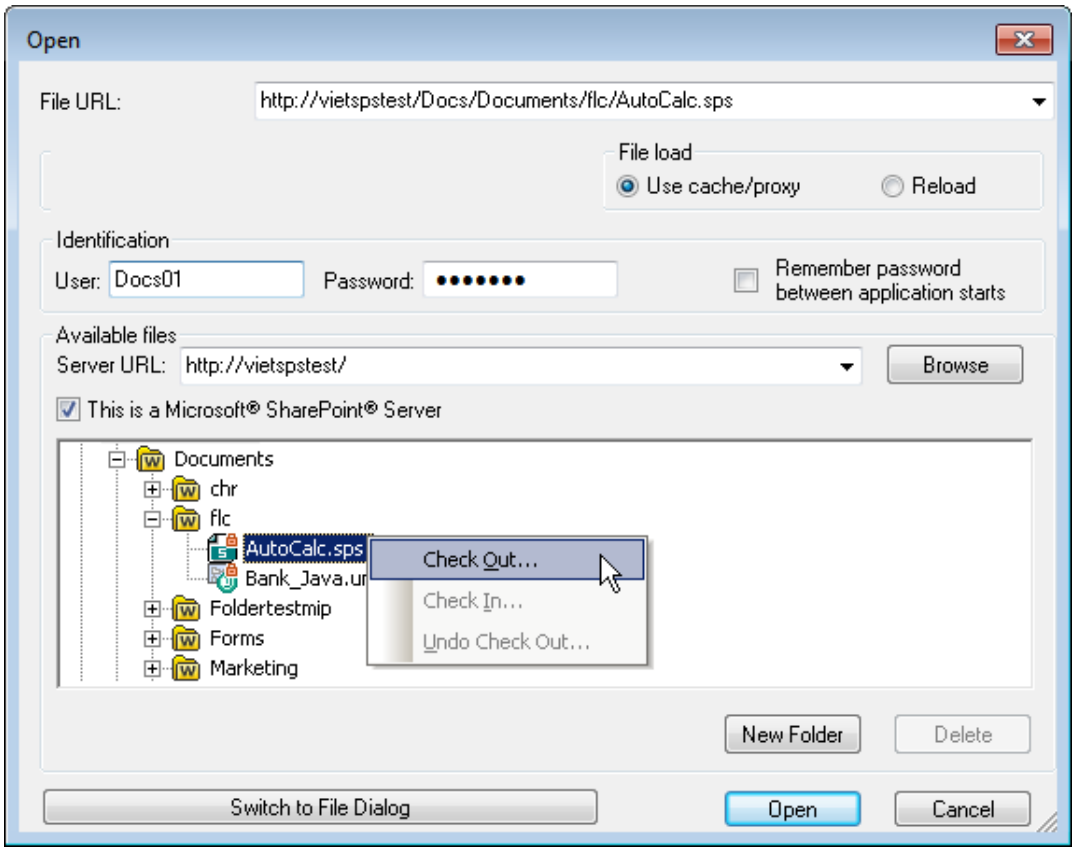
Note the following:

- 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.



Right-clicking a file pops up a context menu containing commands available for that file (*screenshot above*).

- The various file icons are shown below:

	Checked in. Available for check-out.
	Checked out by another user. Not available for check-out.
	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*).

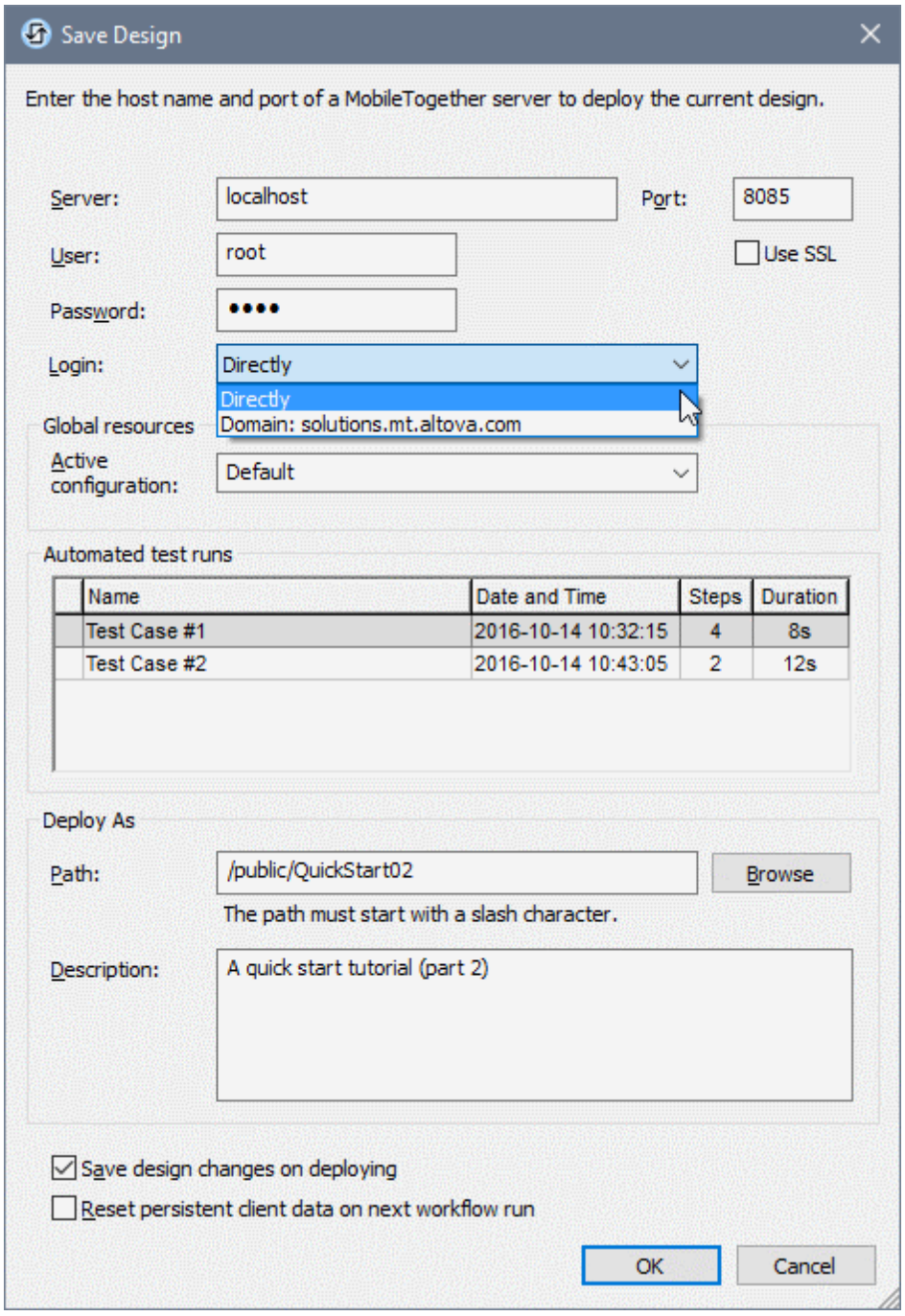
- When a file is checked out by another user, it is not available for check out.
- When a file is checked out locally by you, you can undo the check-out with the Undo Check-Out command in the context menu. This has the effect of returning the file unchanged to the server.
- If you check out a file in 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**.

Deploy to MobileTogether Server

- Deploy to MobileTogether Server
  - *Icon*

  - *Description*

    Opens the Deploy to MobileTogether Server dialog, in which you specify: (i) server connection details, and (ii) deployment properties of the project on the server. On clicking **OK**, the currently active MobileTogether design file is deployed to MobileTogether Server.



The following details are required for deployment:

- *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 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.*
- *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 for details.
- *Automated test cases:* Displays the recorded base test runs (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.
- *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.
- *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

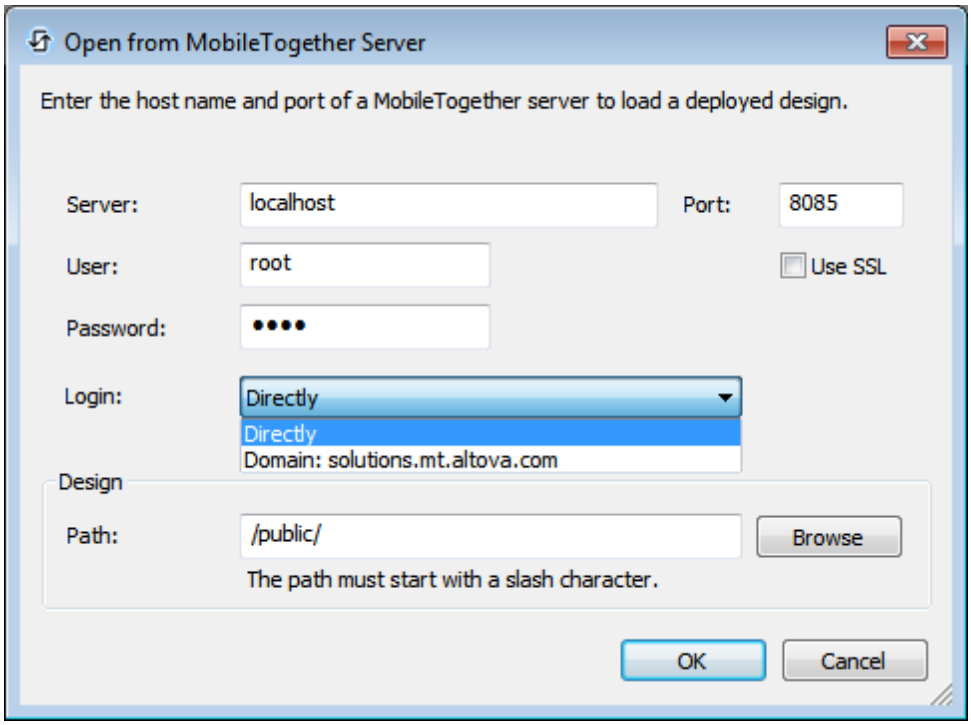
Location of Project Files
Files Pane
Deploying the Project
Data Storage on Servers.

Open from MobileTogether Server

- 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.



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.*

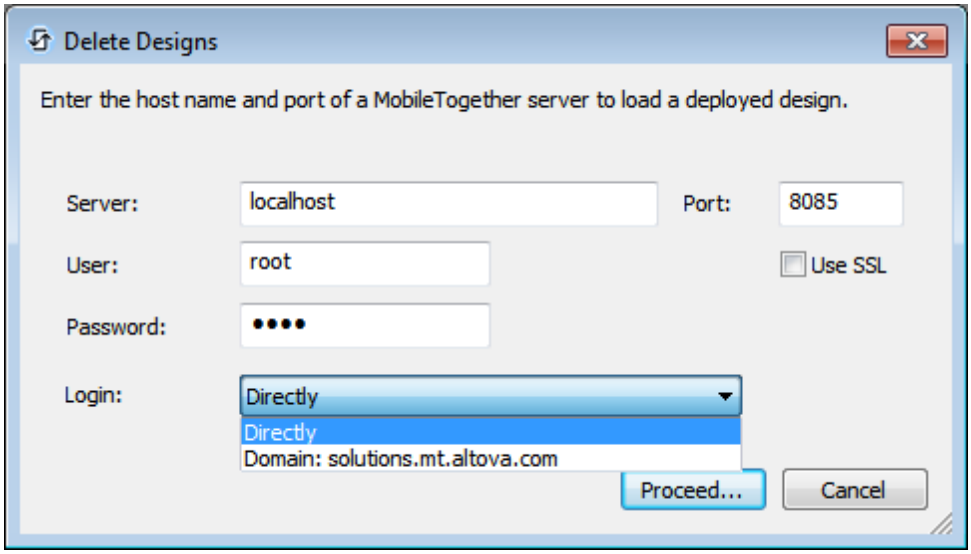
- *Path and name of solution (design):* The path and name of the deployed solution. Click **Browse** to browse through all deployed solutions on MobileTogether Server.

Delete from MobileTogether Server

- 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.

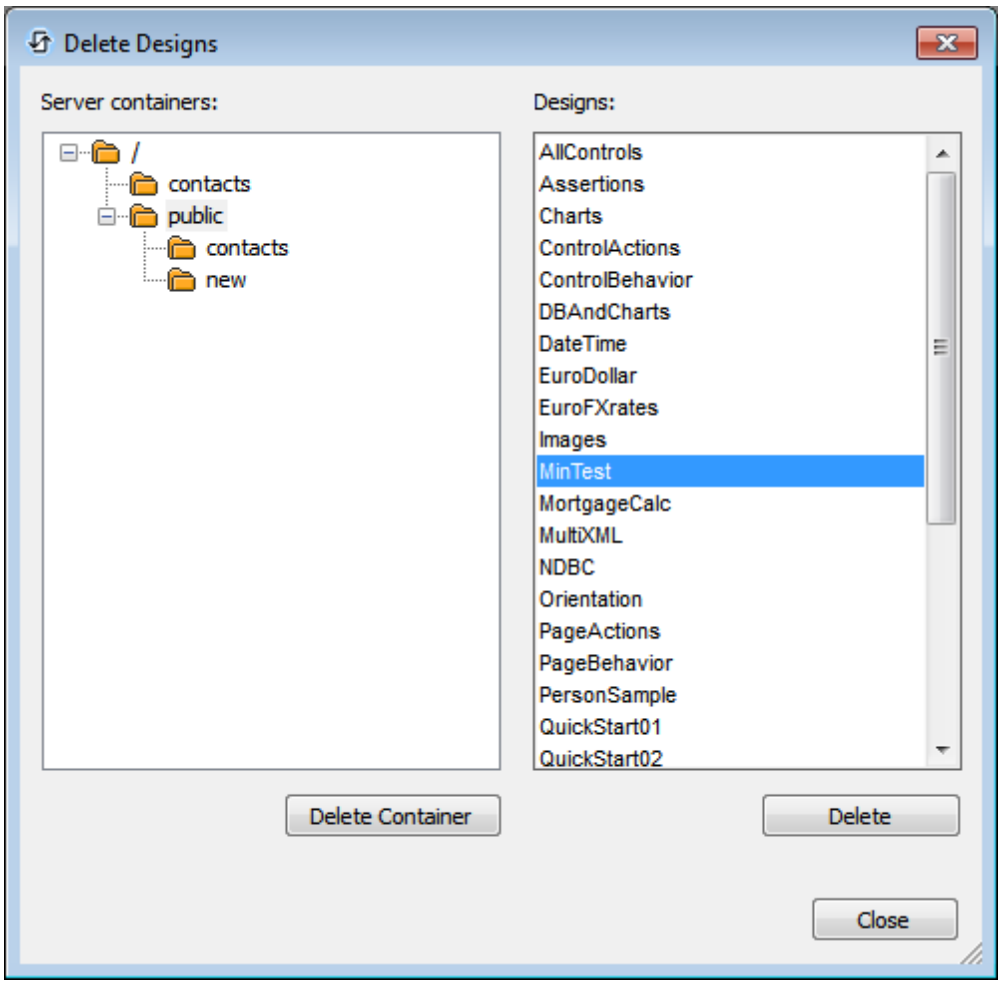


The following details are required for connecting to the server:

- *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

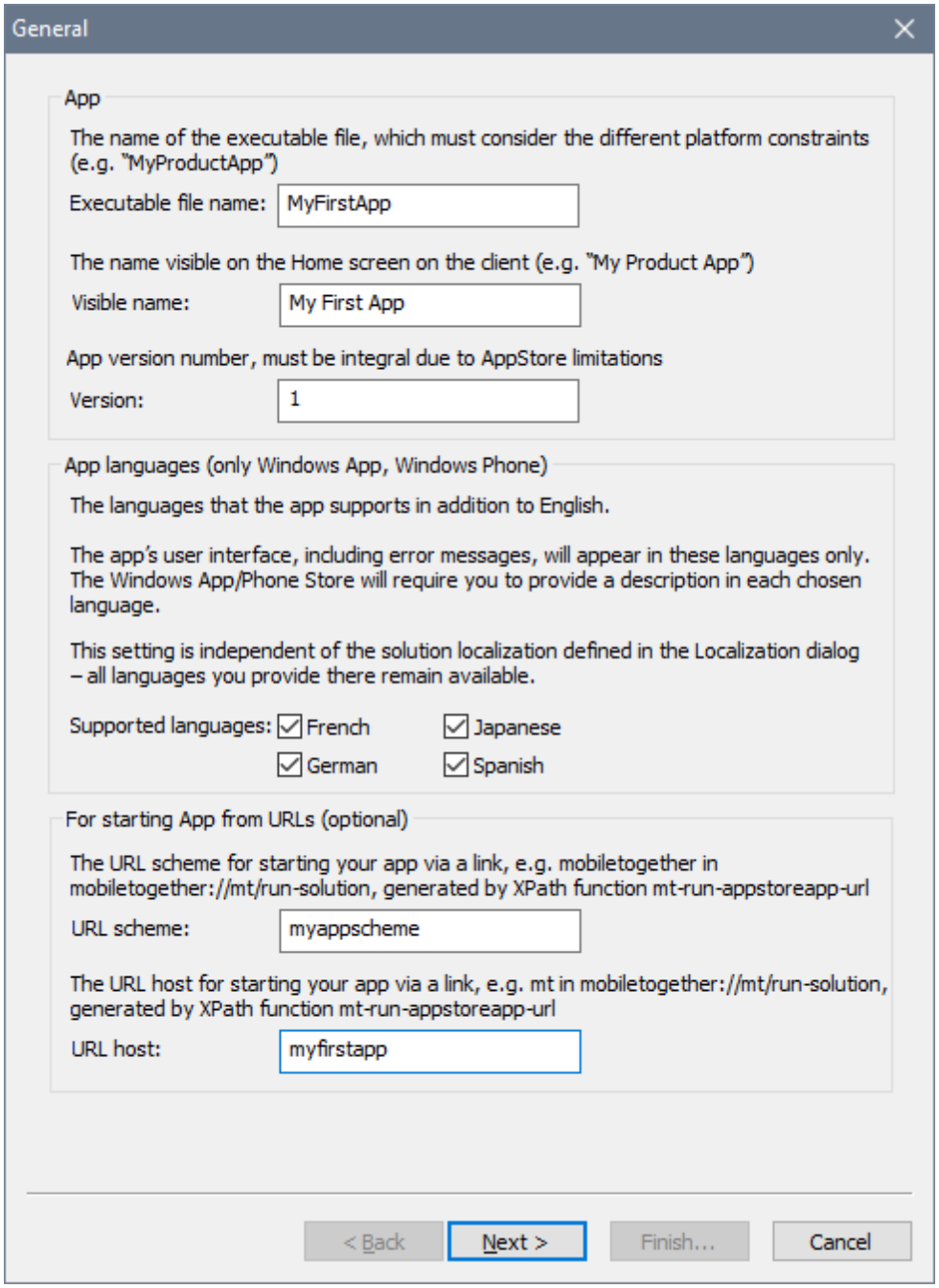
design you wish to delete, select it, and click **Delete Container** or **Delete**.



Generate Program Code for AppStore Apps

- Generate Program Code for AppStore Apps
  - *Description*

    Opens the Generate Program Code Wizard for creating AppStore Apps (*screenshot below*). For a detailed description of how to use the wizard and how to generate appstore apps, see the section AppStore Apps.



After you finish the wizard (by clicking **Finish**), the Deploy Workflow to Server dialog

appears. Clicking **OK** in this dialog (i) deploys the workflow to the server, and (ii) generates program code for the selected app formats (selected in the Generate Program Code Wizard.).

Send by Mail

- 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.

Print

- Print
  - *Icon*

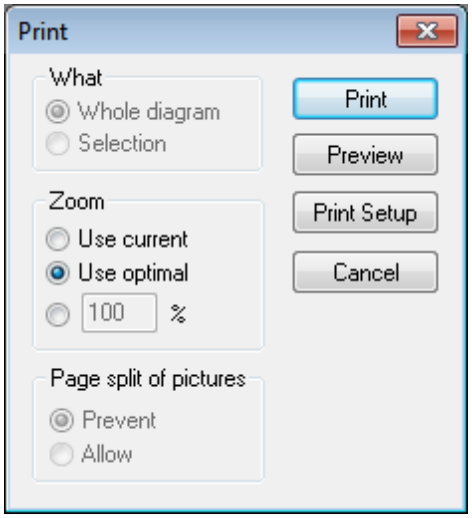
    

  - *Shortcut*

    Ctrl+P

  - *Description*

    Opens the Print dialog box (*screenshot below*), in which you can select printing options for printing the currently active document.

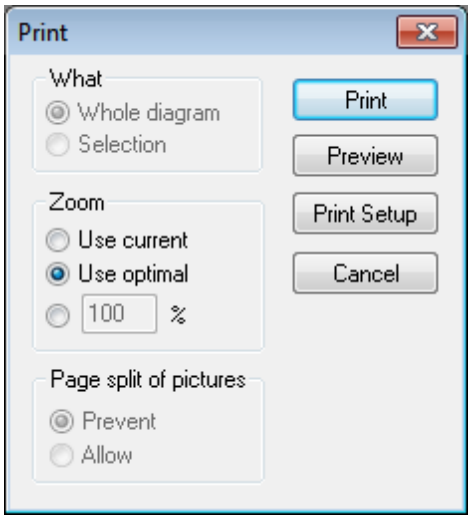
    Options that are applicable to the current printing job are enabled. The following options are available:

    - *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*).

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.

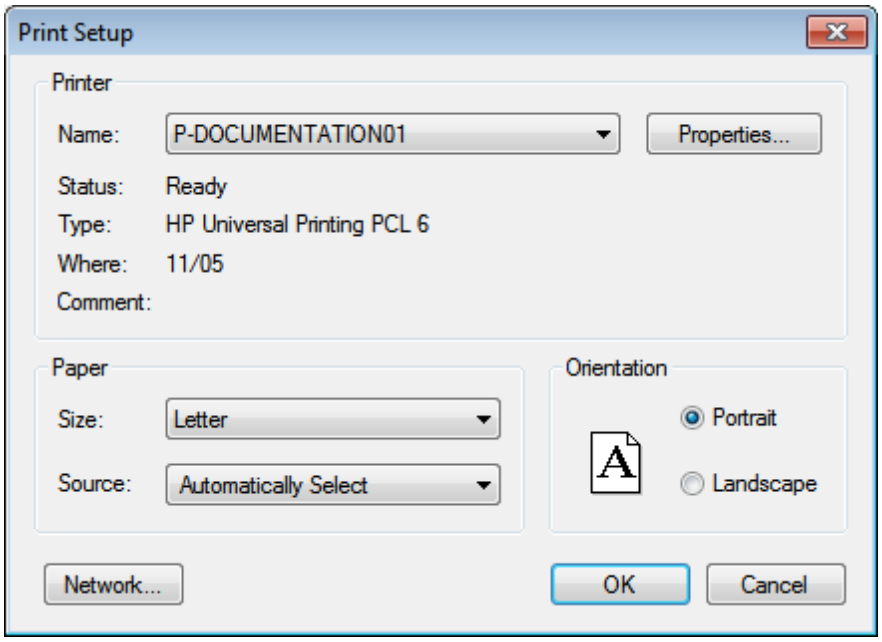


    In Print Preview mode, the Print Preview toolbar at top left of the preview window provides print-related and preview-related options. The preview can be magnified or miniaturized using the **Zoom In** and **Zoom Out** buttons. When the page magnification is such that an entire page length fits in a 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.



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.

20.2 Edit

The **Edit** menu contains the following commands:

- Undo
- Redo
- Cut
- Copy
- Paste
- Delete
- Select All

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.

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.

Select All

- Select All
  - *Shortcut*

    Ctrl+A

  - *Description*

    Selects the contents of the entire document.

20.3 Project

The **Project** menu contains commands that apply to the entire project. It contains the following commands:

- Validate
- Reload Page Source Structures
- Simulate Workflow
- Trial Run on Client
- Use Server for Workflow Simulation
- Record New Test Case
- Playback Test Case
- Trial Run Test Cases on Client
- Manage Test Cases and Runs
- Global Variables
- List Usages of All Global Variables
- List Usages of All Page Source Variables
- XPath/XQuery Functions
- List Usages of All User-Defined XPath/XQuery Functions
- Action Groups
- List Usages of All Action Groups
- Style Sheets
- List Usages of All Style Sheets
- Rich Text Style Sheets
- Cache Overview
- Localization
- Simulation Language
- List All File and Directory References
- List All External Data References
- List Unused Functions, User Variables, Style Sheets, and Action Groups
- Maintain OAuth Settings
- Import OAuth Settings
- iOS Push Notification Button Sets

Validate

- 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 in terms of the number of errors and warnings. If an error or warning is detected, a message about each is displayed.

Reload Page Source Structures

- Reload Page Source Structures
  - *Icon*

  - *Description*

    Reloads the page source data structures that have been defined in the Page Sources Pane. Use this command to refresh the page's source data while designing the page.

Simulate Workflow

- Simulate Workflow
  - *Icon*

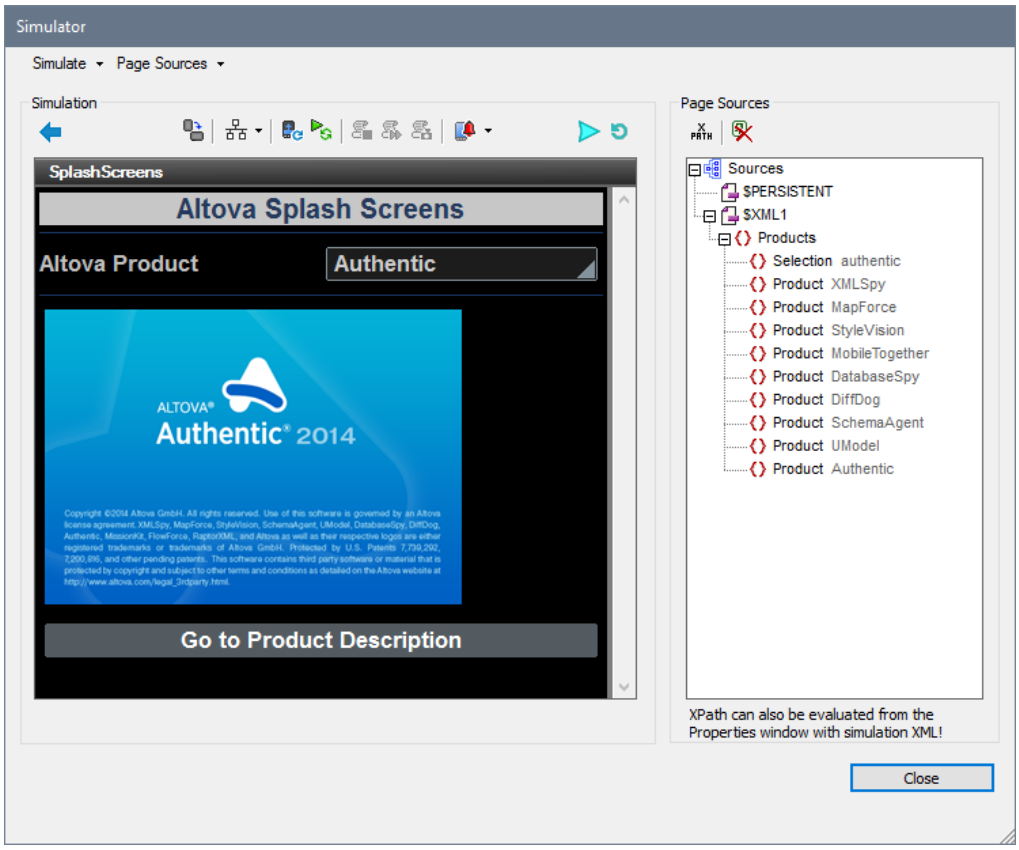
    

  - *Shortcut*

    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 will be simulated.



Trial Run on Client

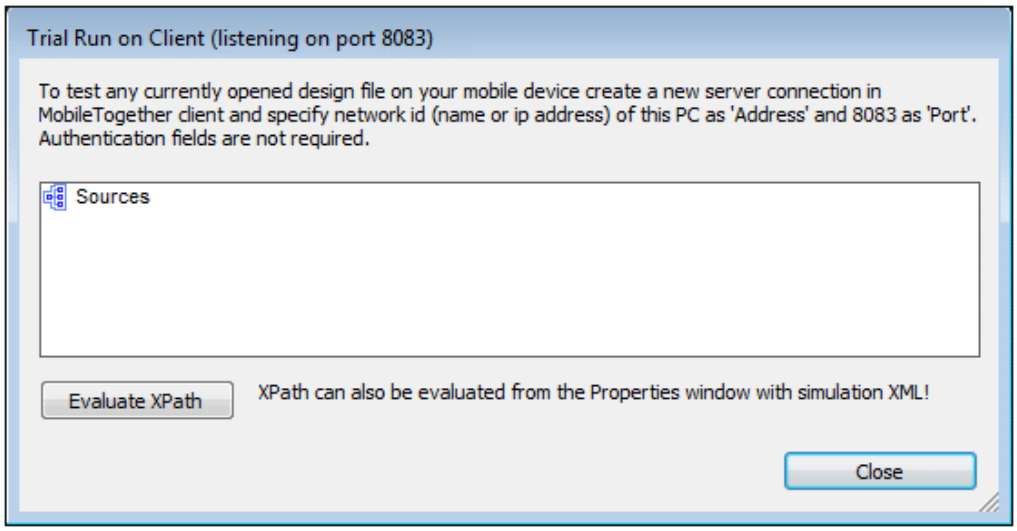
- Trial Run on Client
  - *Icon*

  - *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 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.



Use Server for Workflow Simulation

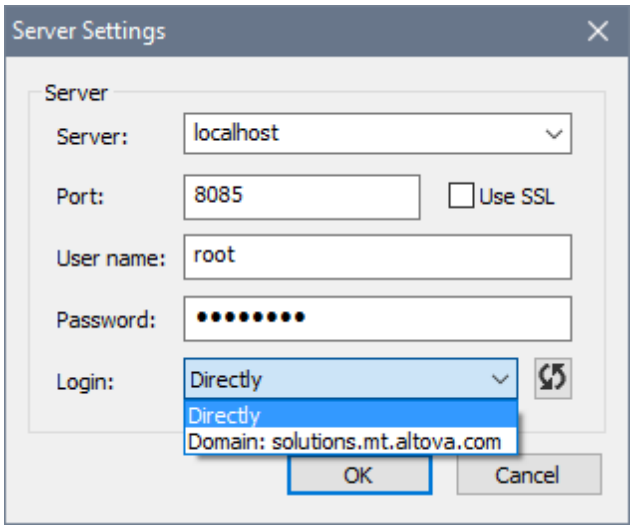
- Use Server for Workflow Simulation
  - *Icon*

  - *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 if this has been set up (*see the description of the Server Settings tab of the Options dialog*). On clicking **OK**, the simulation starts in a separate window.



Record New Test Case

- Record New Test Case
  - *Icon*

  - *Description*

    Starts a new test case in the Simulator 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* dialog. *See Recording a Test Case for details*.

Playback Test Case

- 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 for details*. Note that options for playback are specified in the *Manage Test Cases and Runs* dialog.

Trial Run Test Cases on Client

- Trial 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 for details*.

Manage Test Cases and Runs

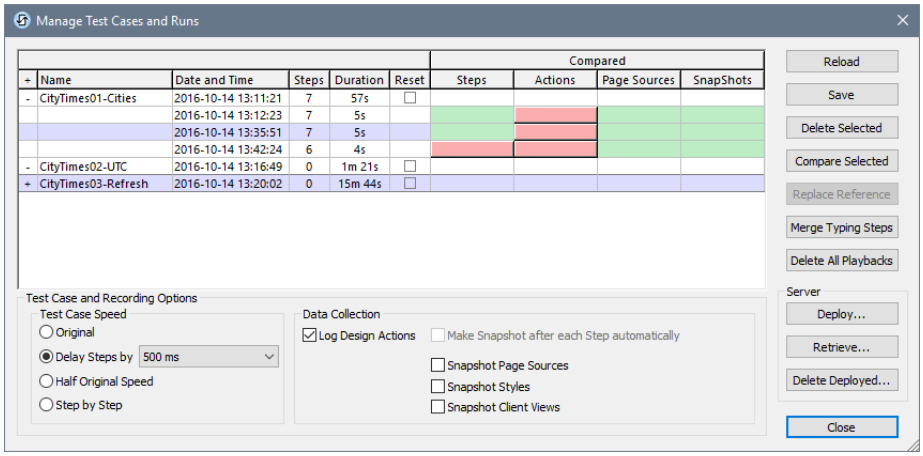
- Manage Test Cases and Runs
  - *Icon*

  - *Description*

Displays the *Manage Test Cases and Runs* 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.



See Managing Test Cases and Runs for details.

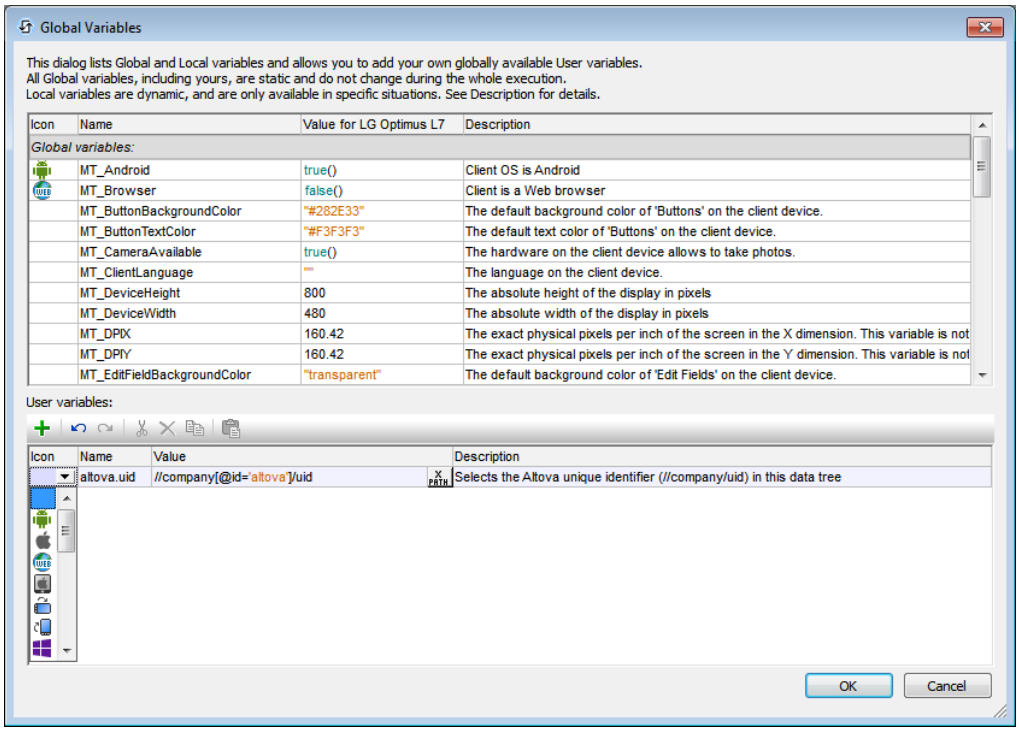
Global Variables

- 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. 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.



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 for a description of predefined global variables.

List Usages of All Global Variables

List Usages of All Global Variables

Description

Returns a list, in the Messages Pane, of all global variables. 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.

List Usages of All Page Source Variables

List Usages of All Page Source Variables

Description

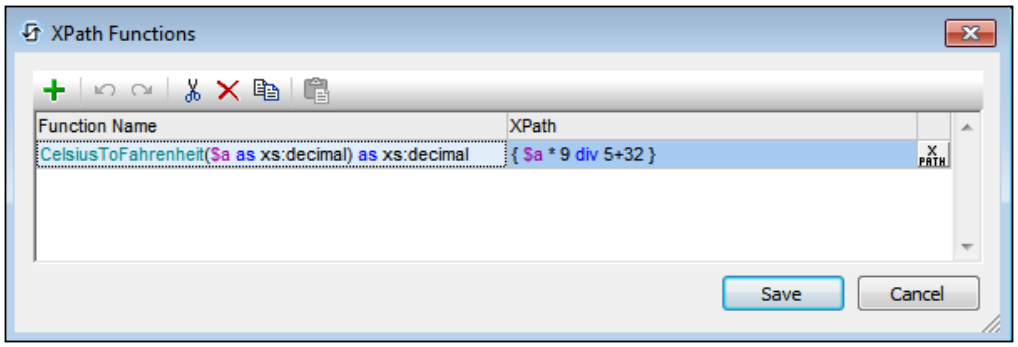
Returns a list, in the Messages Pane, of all page source variables, each of which indicates a page source. Each page source variable contains links to the definitions in which that variable is used. This enables you to quickly locate and edit that definition. Clicking the page source link itself highlights the page source in the Page Sources Pane.

XPath/XQuery Functions

- XPath/XQuery Functions
  - *Icon*

  - *Description*

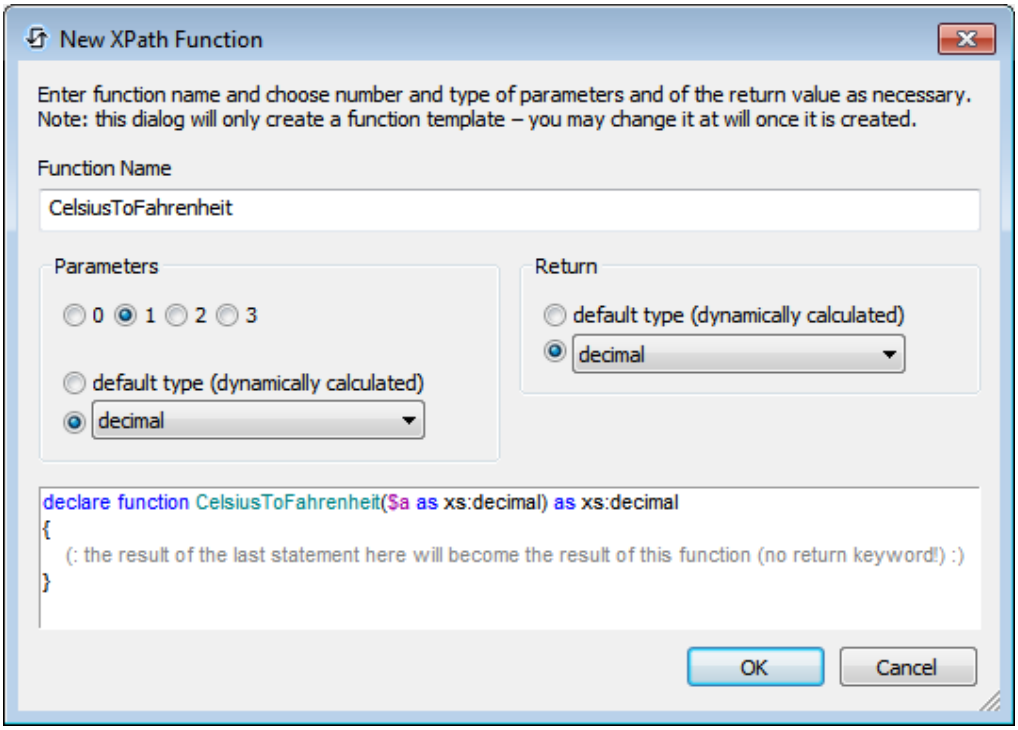
Opens the XPath Functions dialog, which lists all the user-defined XPath functions in the project. These XPath functions can be used in all XPath expressions in the project. You can add and delete functions using the corresponding icons in the toolbar of the dialog. To edit the definition of a function, click the function's **Edit XPath Expression** button.



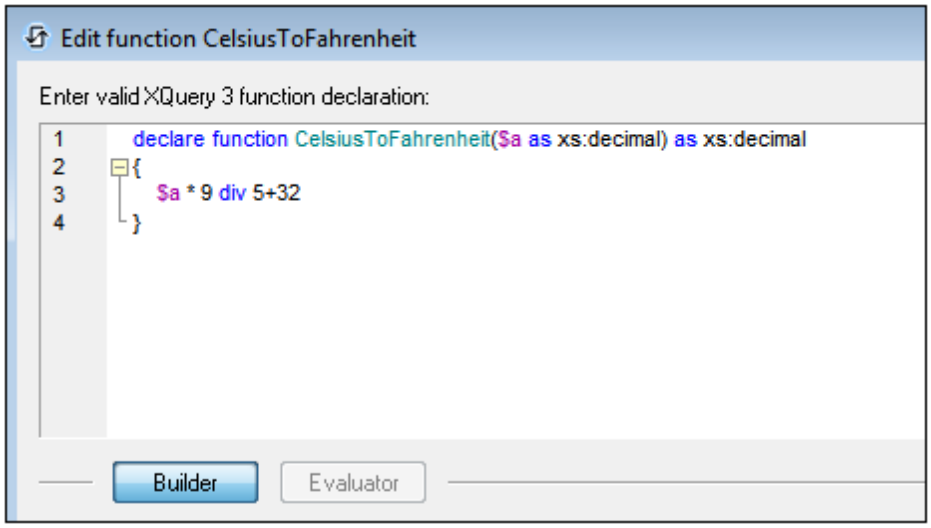
Add a new user-defined XPath function

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*).



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.



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 is used for all XPath/XQuery functions, including extension functions and user-defined functions. In order to avoid ambiguities involving built-in functions, we recommend that you capitalize user-defined functions.

List Usages of All User-Defined XPath/XQuery Functions

List Usages of All User-Defined XPath/XQuery Functions

Description

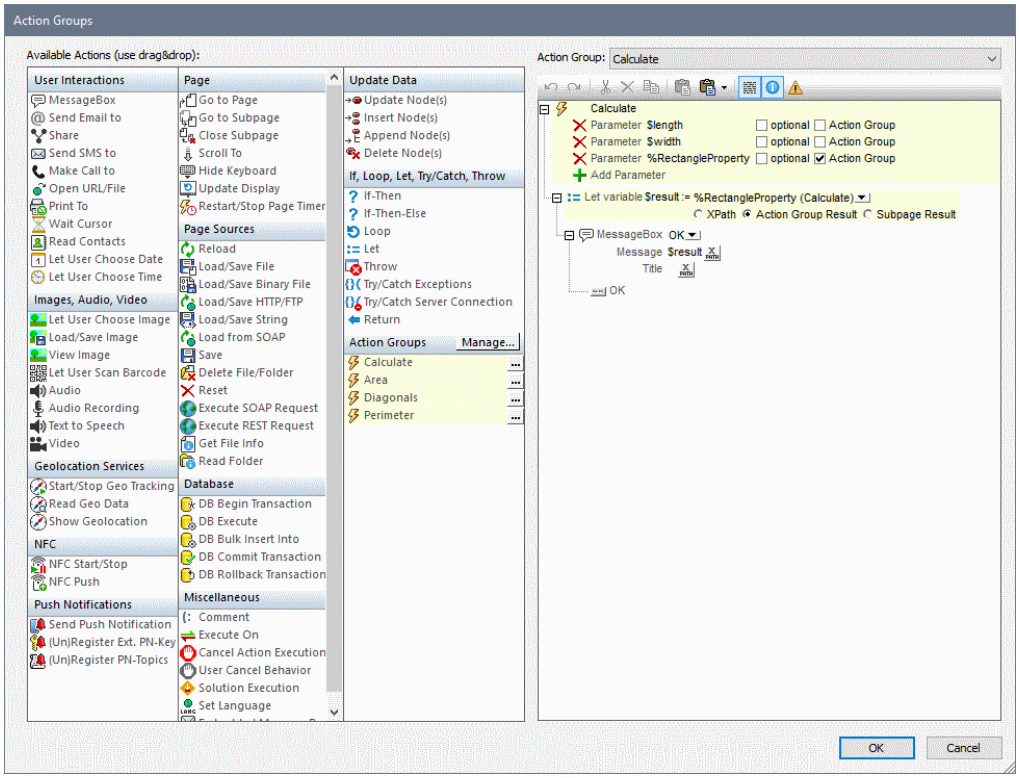
Returns a list, in the Messages Pane, of all the user-defined XPath/XQuery functions used by pages in the project, together with the pages in which they occur. Clicking the links in the listing takes you directly to the dialog defining the XPath/XQuery function or the definition containing the XPath/XQuery function.

Action Groups

- Action Groups
  - *Icon*

  - *Description*

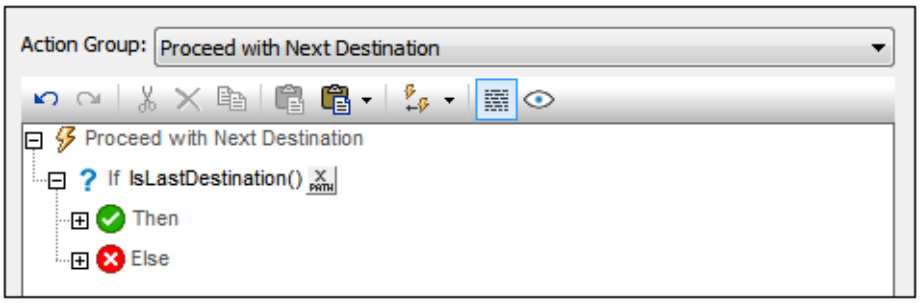
Displays the Action Groups dialog (*screenshot below*), in which action groups can be created and edited.



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*).



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.

List Usages of All Action Groups

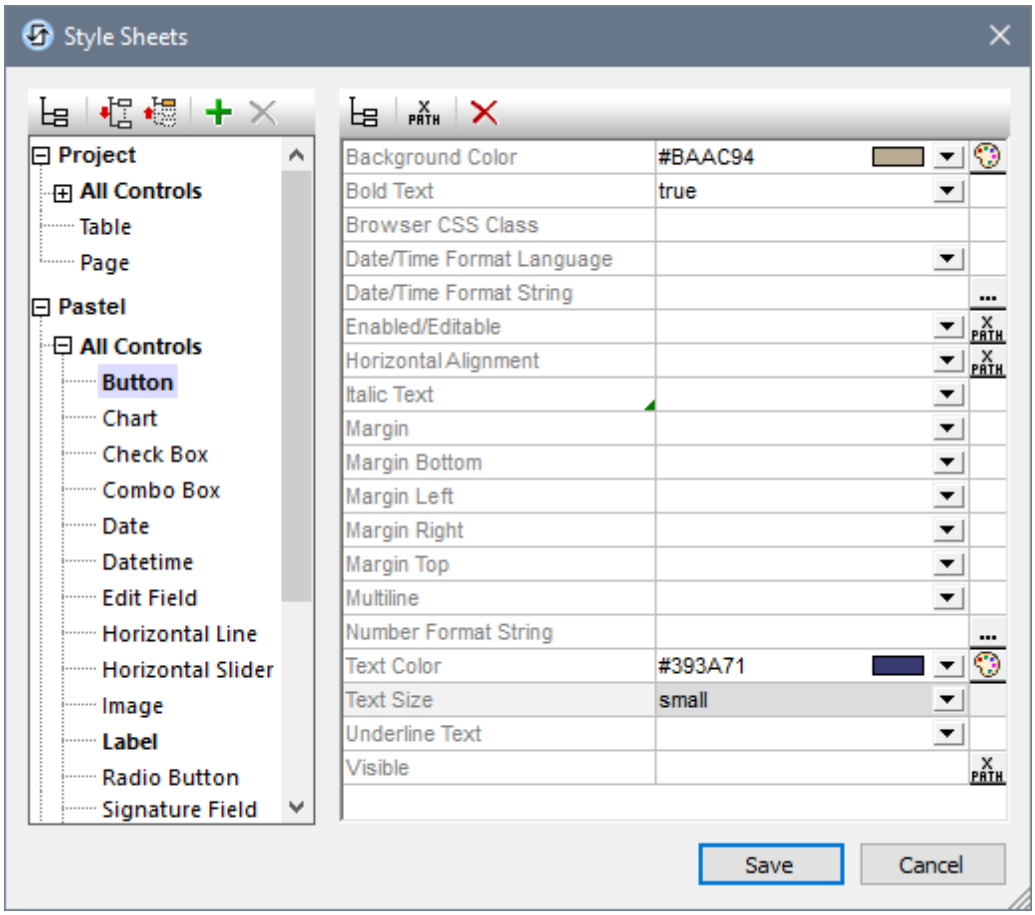
- List Usages of All Action Groups
  - *Description*

    Returns a list, in the Messages Pane, 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.

Style Sheets

- Style Sheets
  - *Description*

Displays the Style Sheets dialog (*screenshot below*), in which you can manage style sheets and define the styles in the different style sheets.



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 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 for more information*.

Click **Save** when you are done.

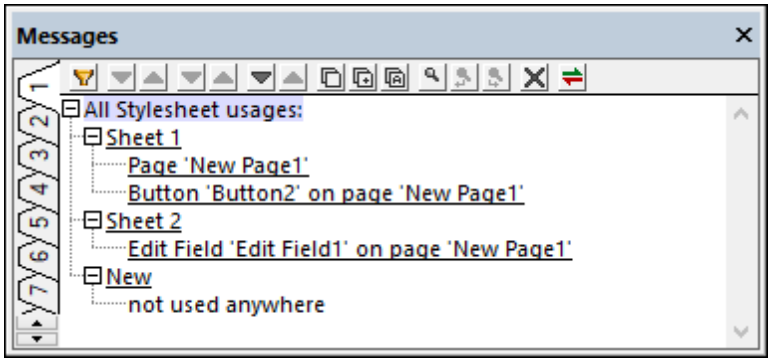
Note: Each pane has an icon that enables/disables the display of non-empty items (that is, those items for which a value (or at least one value) has been defined). Displaying only the non-empty items is useful when you wish to see a list of only the styles that have been defined; for example, when you wan t 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.

List Usages of All Style Sheets

- List Usages of All Style Sheets
  - *Description*

A user-created style sheet can be applied to page instances, table instances, and control instances. This command returns a list, in the Messages Pane, of all the user-created style sheets in the project and where each is used (*see screenshot below*).



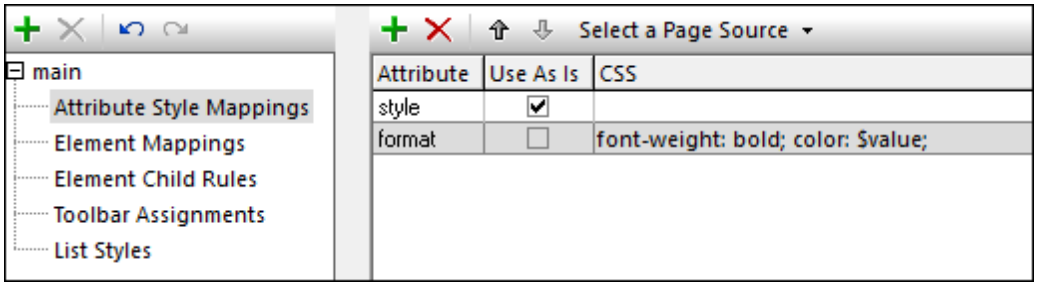
The list is ordered by style sheet. Under each style sheet, all the page, table, and control instances are listed that use that style sheet. Clicking one of these items takes you directly to the respective page, table, or control.

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** command of the **Project** menu.

Rich Text Style Sheets

- Rich Text Style Sheets
  - *Description*

    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



For a description of the Rich Text Style Sheets dialog, see the Rich Text section.

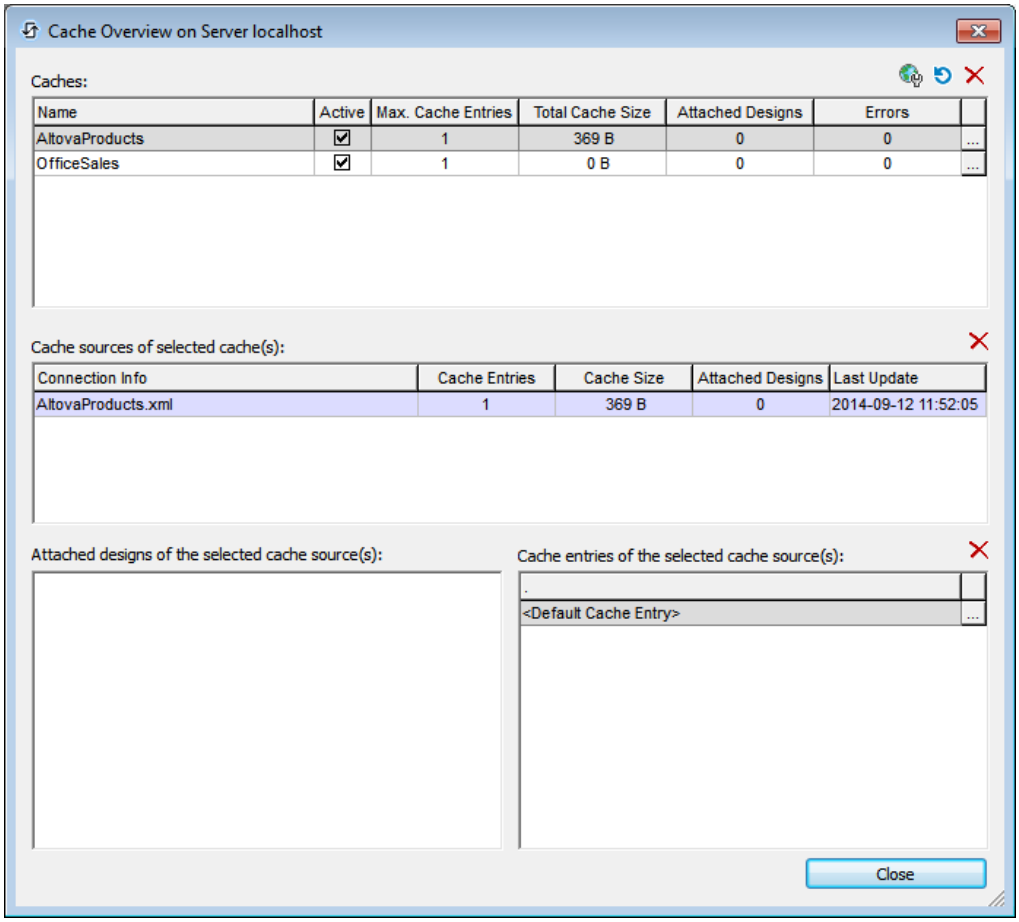
Cache Overview

- Cache Overview
  - *Icon*

  - *Description*

Enables connection to MobileTogether Server and an overview of the server cache: cache hits, cache size, purges, etc.

The Cache Overview dialog (*screenshot below*) is accessed with the menu command **Project | Cache Overview**.



The dialog provides an overview of all the caches on the server. In it, you can also do the following:

- Activate/deactivate caches.
- Delete caches.

See the section Caching for more information about caching.

Localization

- Localization
  - *General description*

    Displays the Localization dialog (*screenshot below*), in which you can localize (translate) strings that 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` function.

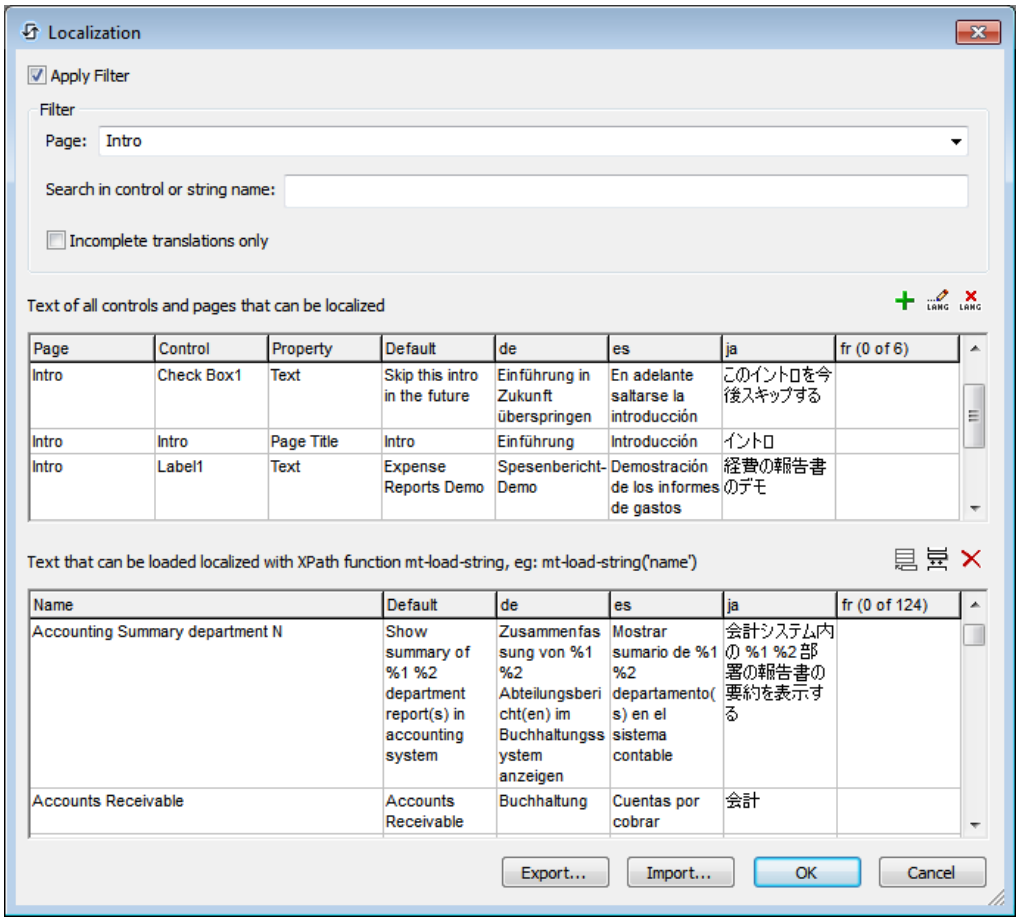
    - **Control-related strings** are displayed in the upper pane (*see screenshot below*). The columns of this pane are, respectively, from left to right: (i) the page in which the control appears; (ii) the control's name; (iii) the control property that contains the text string; (iv) the text string in 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.

    You can add a column for a localization language with the **Add Language** icon, and 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** command, and then run a simulation.



- *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.

- *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`** *XPath function.*)
	**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.
	**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.

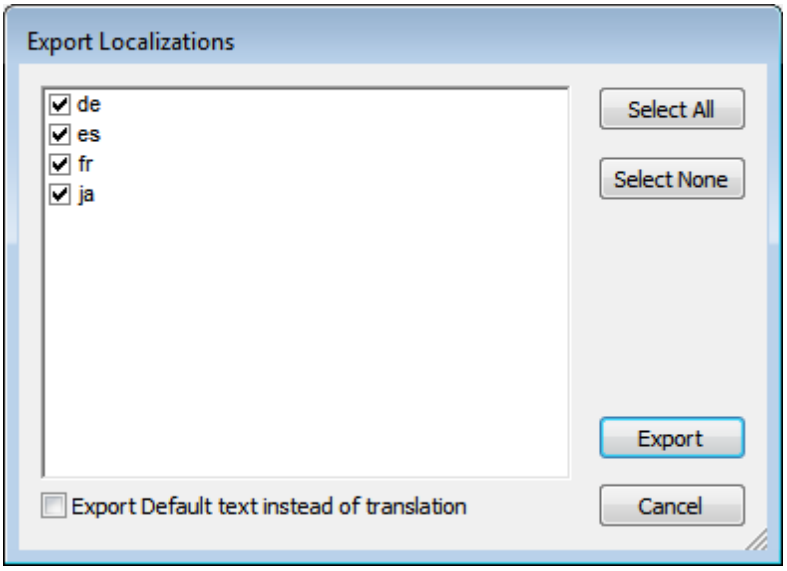
	**Add Localization String**	Adds a row for a custom string. The newly added custom string is created with a default name, which you can edit. Click in the name to start editing. The name is used to reference the custom string from XPath expressions. *See below.*
	**Duplicate Row**	Duplicates the custom 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`** function in an XPath expression, like this: **`mt-load-string('NameOfString')`**.

- *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="Administration"/>
    </String>
    ...
  </Strings>
</Localizations>
```

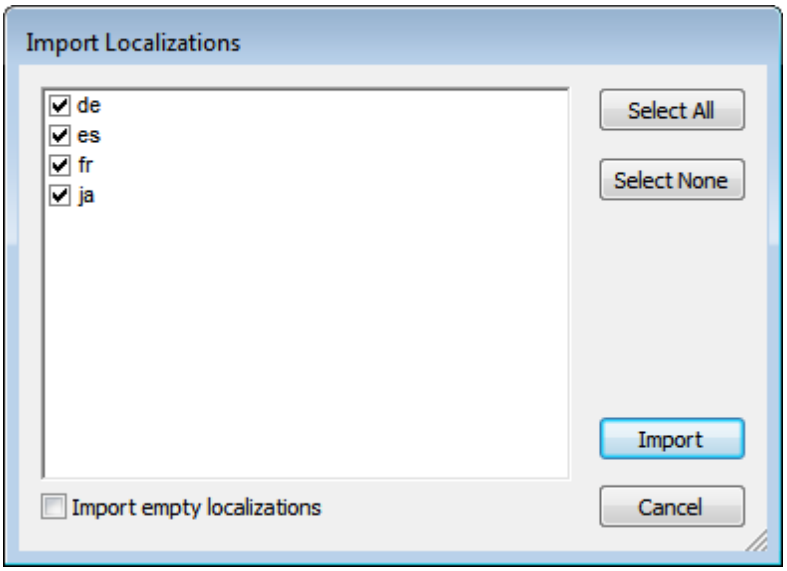
XML listing of multiple-language export

```
<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.



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 these are non-empty. 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.

Simulation Language

- Simulation Language
  - *Description*

    When a project is localized, 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 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.

List All File and Directory References

- List All File and Directory References
  - *Description*

    Returns a list, in the Messages Pane, 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.

List All External Data References

List All External Data References

Description

Returns a list, in the Messages Pane, 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.

List Unused Functions, User Variables, Style Sheets, Action Groups

List Unused Functions, User Variables, and Action Groups

Description

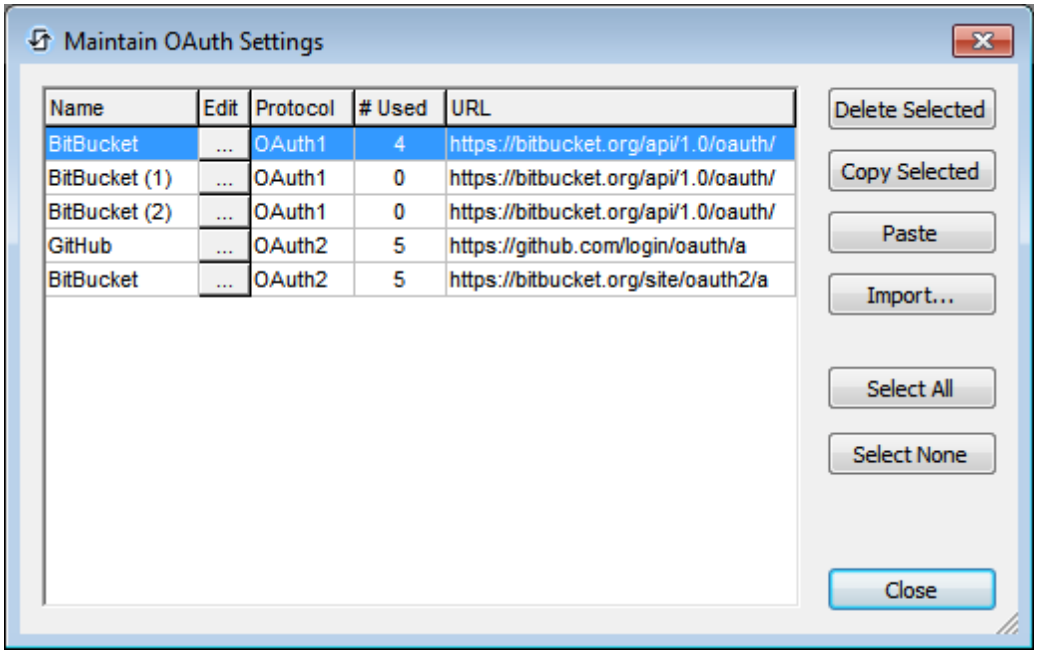
Returns a list, in the Messages Pane, of all the **unused** XPath/XQuery functions, user variables, style sheets, and action groups that are defined in the project. This is useful if you wish to review these user-defined components that are not in use and clean up the project. Clicking the links in the listing takes you directly to the respective definition.

Maintain OAuth Settings

- Maintain OAuth Settings
  - *Description*

REST requests made in MobileTogether Designer can be authenticated with OAuth. See the section REST Request Settings 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 Maintain OAuth Settings dialog (*screenshot below*) lets you manage the OAuth definitions of the active project. The dialog displays all the OAuth settings definition 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.



The dialog has the following columns:

- *Name:* The name that was assigned to the settings definition when it was created. The name cannot be edited.
- *Edit:* The button opens the OAuth Settings dialog, 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.

You can carry out the following actions in this dialog:

- *Delete Selected:* One or more definitions can be selected for deletion..

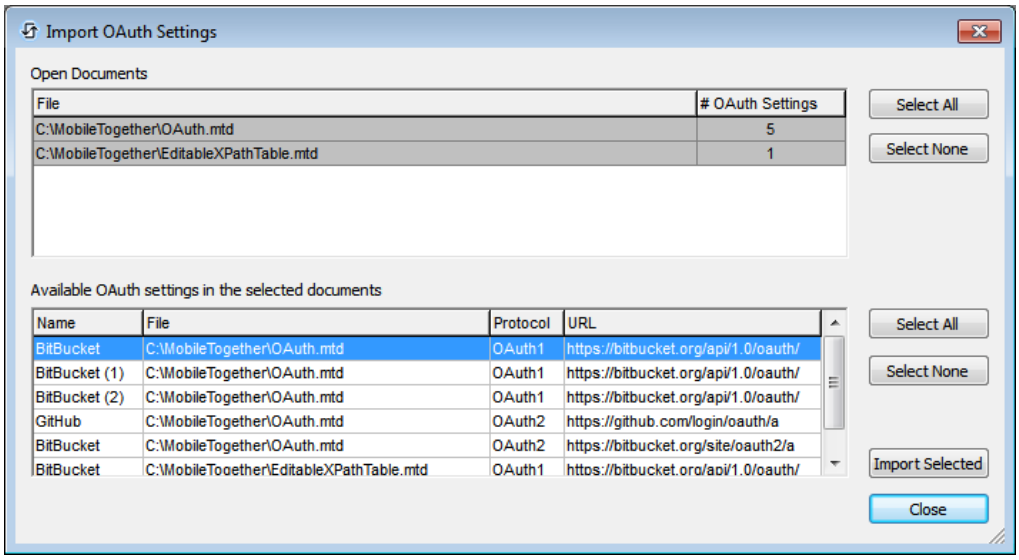
- *Copy Selected:* Copies one or more definitions as a group to the clipboard.
- *Paste:* This button is enabled when one or more OAuth settings definitions is currently in the clipboard. Copies the clipboard contents to the current project's pool of definitions
- *Import:* Opens the **Import OAuth Settings** dialog, which enables you to import one or more OAuth definitions from other open MobileTogether Designer files. See the **Import OAuth Settings** dialog for details.
- *Select All*: Selects all the definitions.
- *Select None*: Selects none of the definitions.

Import OAuth Settings

- Import OAuth Settings
  - *Description*

REST requests made in MobileTogether Designer can be authenticated with OAuth. See the section REST Request Settings 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.



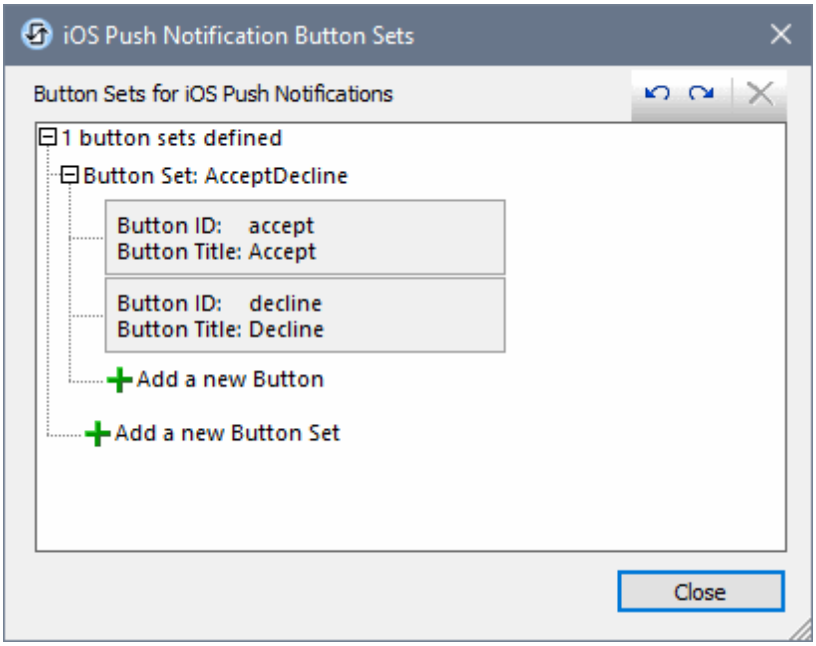
The Open Documents pane (*see screenshot above*) displays all the other documents that are currently open in MobileTogether Designer. Select one or more documents to display their OAuth settings definitions in the lower pane. In the lower pane, select one or more definitions that you want to import into the current MobileTogether Designer project, and then click **Import Selected**. The definitions will be imported and can be viewed in the Maintain OAuth Settings dialog.

iOS Push Notification Button Sets

- iOS Push Notification Button Sets
  - *Description*

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, 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 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.

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.

See Push Notifications for an overview and more information about PNs.

20.4 Page

The **Page** menu contains commands that apply to the currently active page of a project. It contains the following commands:

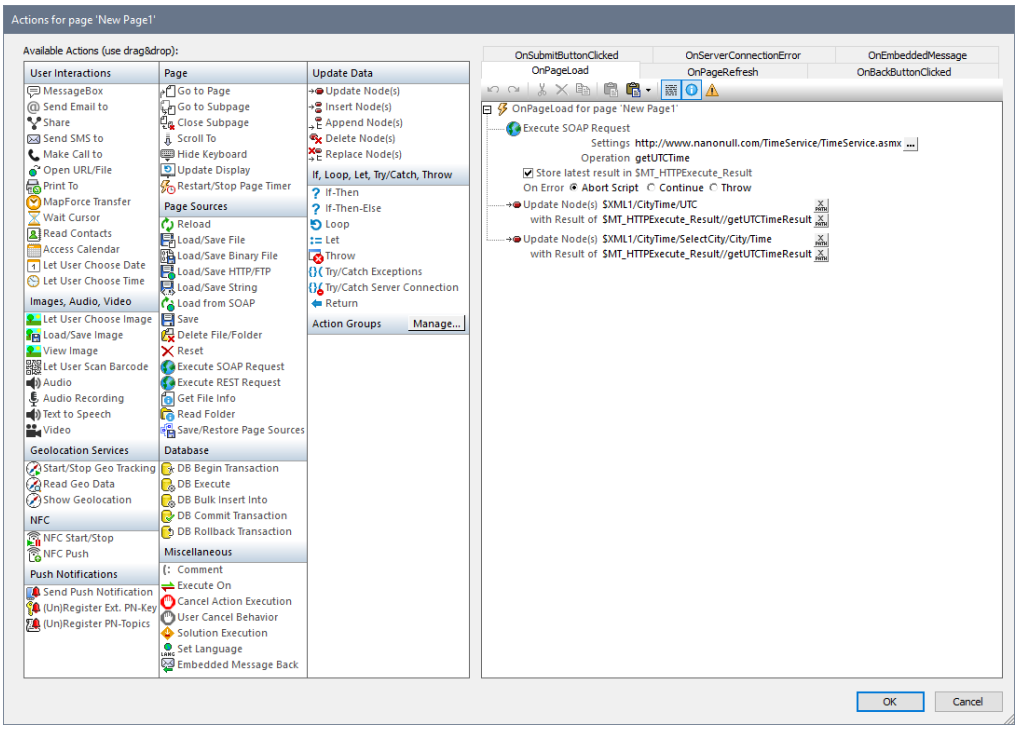
- Page Actions
- Actions Overview
- Jump to Control
- Show/Define Tab Order

Note: **Page** menu commands are enabled only in Page Design View.

Page Actions

- Page Actions
  - *Description*

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.

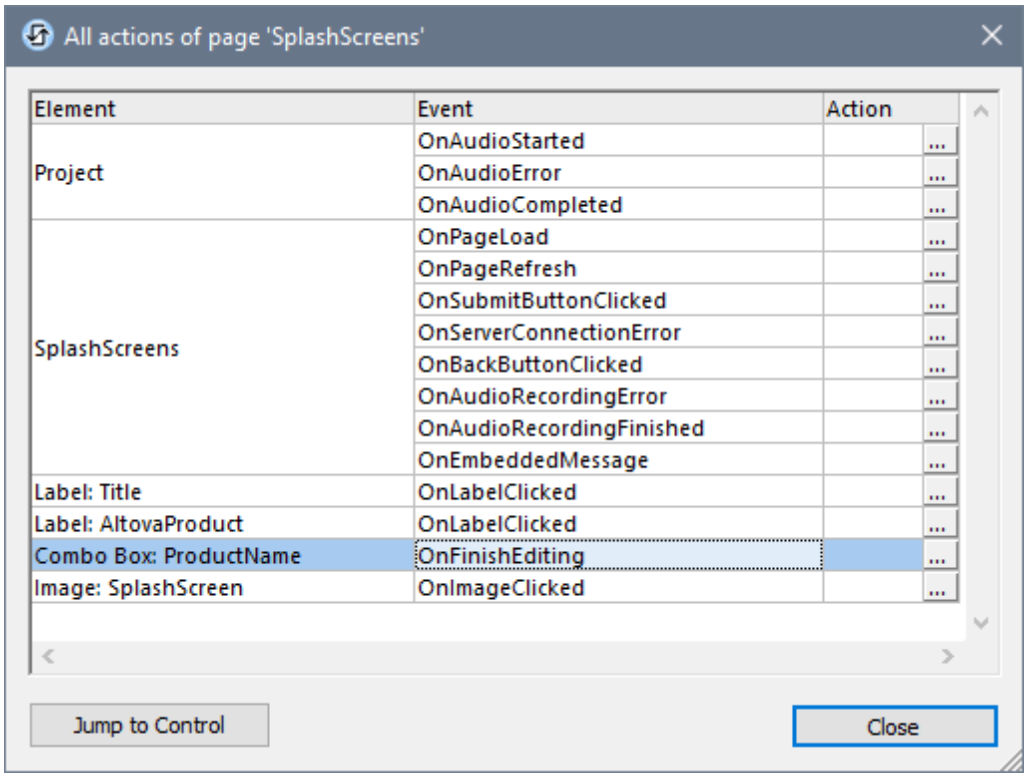


To specify that a particular action (from among the available actions) is carried out when an available event occurs, drag the action from the left-hand pane into the event's tab in the right-hand pane. Specify additional properties of the action as required. (For more information about individual page actions, see the section, Page Actions.) Note that you can also add any Action Group that might be defined for the project. Click **OK** to finish.

Actions Overview

- Actions Overview
  - *Description*

Displays the Actions Overview dialog of the currently active page (*see screenshot below*). The dialog shows Control Actions and Page Actions. Each control in the page design is listed, together with its event/s and corresponding action/s; these are Control Actions. The dialog also shows available events for the active page; these are Page Actions. For instance, in the screenshot below, the `Income` item is a page item: it has page events that take page actions. All the other items are controls: they have control events that take control actions.



- 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 Actions dialog of that control or to the Actions dialog of the page.
- When a control is selected (not the page), 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.

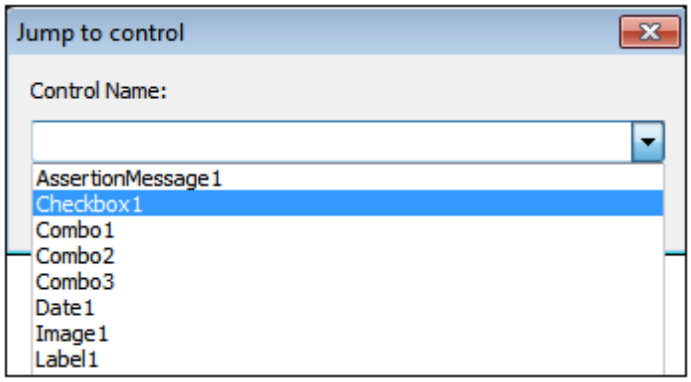
Jump to Control

- 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 will also be selected.

Show/Define Tab Order

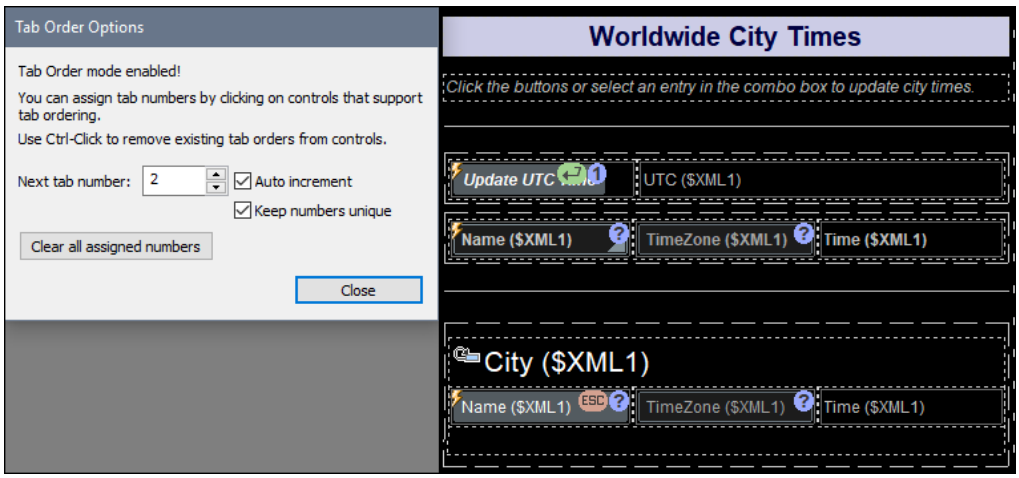
- Show/Define Tab Order
  - *Shortcut*

    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*).



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.

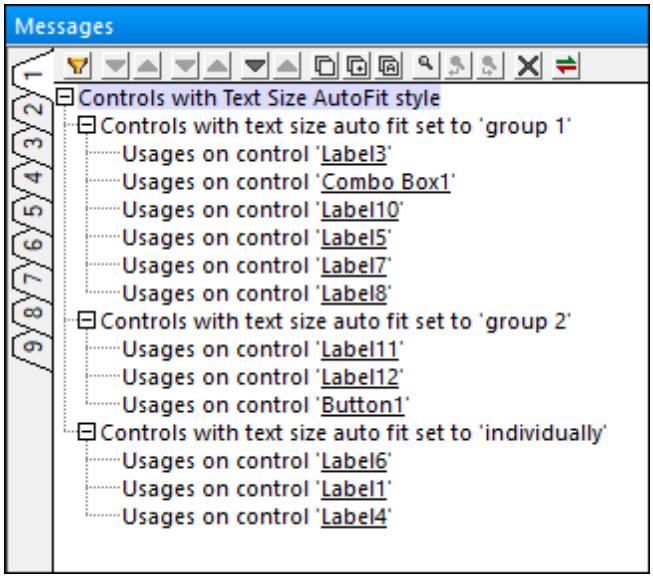
Displaying the tab sequence, indicating controls assigned Enter/Escape shortcuts
When the Tab Order Options dialog is opened, the following indicators in the design become visible:

- 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.

List Text Size Auto-Fit Groups

- List Text Size Auto-Fit Groups
  - *Description*

    Displays in the Messages Pane 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 for which the property is available, for example, the Label or Button control.



20.5 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
- Delete Table
- Insert Row
- Append Row
- Delete Row
- Insert Column
- Append Column
- Delete Column
- Join Cell Left
- Join Cell Right
- Join Cell Above
- Join Cell Below
- Split Cell Horizontally
- Split Cell Vertically
- 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

For more information about tables, see the section Design Object/Features | Tables.

Insert/Delete Table

The **Insert Table** and **Delete Table** commands are available when a table cell in any kind of table (static, repeating, or tables with dynamic rows and/or dynamic columns) 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.

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, repeating, or tables with dynamic rows and/or dynamic columns) 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 when a table cell is selected.

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, repeating, or tables with dynamic rows and/or dynamic columns).

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*.

Splitting cells

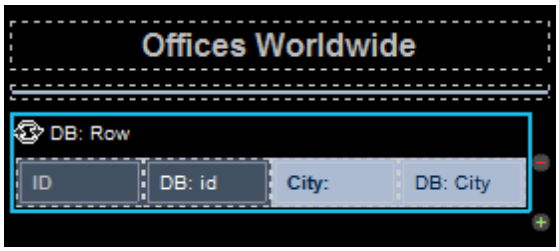
You can split a cell horizontally or vertically into two cells. If a cell is split horizontally, then any content or formatting that was present in the original cell will be retained in the left-hand cell of the new pair; the right-hand cell will be empty. If the cell is split vertically, then the upper cell of the new cell pair will retain the content and formatting of the original cell; the lower pair will be empty.

Show Add–Remove Buttons

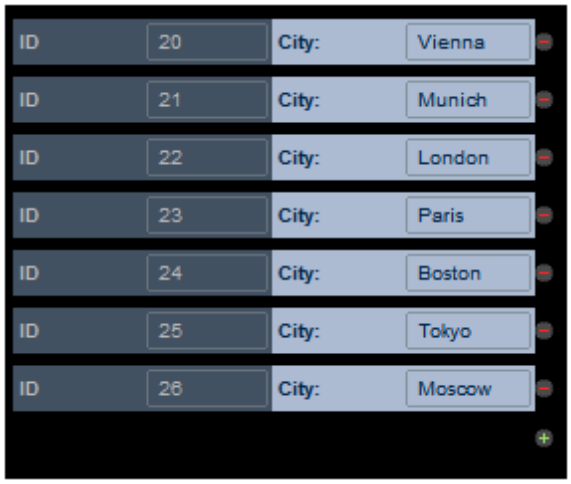
▼ Show Add-Remove Buttons

▣ *Description*

The **Show Add–Remove Buttons** command is available for repeating tables or the repeating rows of tables with dynamic rows. It creates Add–Remove buttons for the selected row. The screenshot below shows the design of a table with dynamic rows 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 data source. You can add a new row by tapping the **Add** button.



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 or dynamic (repeating) columns.

- Insert Table Header
  - *Icon*

  - *Description*

    Inserts an empty header row in tables with dynamic rows 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 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 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 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.)*

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 or dynamic (repeating) columns.

- Remove Table Header
  - *Icon*

  - *Description*

    Removes the header of tables with dynamic rows. 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. 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. 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. 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.)*

Convert Row to Repeating/Static Row

▼ Convert This Row to Repeating Row

▣ *Description*

If the selected row ***is not*** a dynamic (repeating) row, then this command converts it to a repeating row. If the selected row is a repeating row, then this command is disabled.

▼ Convert This Row to Static Row

▣ *Description*

If the selected row ***is*** a dynamic (repeating) row, then this command converts it to a static row. If the selected row is not a repeating row, then this command is disabled.

Convert to Repeating/Non-Repeating Table

- Convert to Repeating Table

  - *Description*

    If the selected table ***is not*** a repeating table, then this command converts it to a repeating table. If the selected table is a repeating table, then this command is disabled.

- Convert to Non-Repeating **Table**

  - *Description*

    If the selected table ***is*** a repeating table, then this command converts it to a non-repeating table. If the selected table is not a repeating table, then this command is disabled.

Convert Column to Repeating/Static Column

▼ Convert this Column to Repeating Column

▣ *Description*

If the selected column ***is not*** a dynamic (repeating) column, then this command converts the column to a repeating column. If the selected column is a repeating column, then this command is disabled.

▼ Convert this Column to Static Column

▣ *Description*

If the selected column ***is*** a dynamic (repeating) column, then this command converts the column to a static column. If the selected column is not a repeating column, then this command is disabled.

20.6 View

The **View** menu contains the following commands:

- Status Bar
- Pages
- Files
- Controls
- Page Sources
- Overview
- Styles & Properties
- Messages
- All On/Off
- Zoom
- Zoom In
- Zoom Out
- Zoom Reset to 100%
- Zoom to Selection
- Fit to Window

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
- Files Pane
- Controls Pane
- Page Sources Pane
- Overview Pane
- Styles & Properties Pane
- Messages Pane

The **All On/Off** command toggles all the panes and the status bar together between being displayed and being hidden.

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.

- 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.

- Zoom In
  - *Icon*

  - *Description*

    Magnifies the diagram by 10 percent points (`100`, `110`, `120...`) each time the command is executed. In Page Design View, 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.

20.7 Tools

The **Tools** menu contains the following commands:

- Global Resources
- Active Configuration
- User-Defined Tools
- Customize
- Restore Toolbars and Windows
- Options

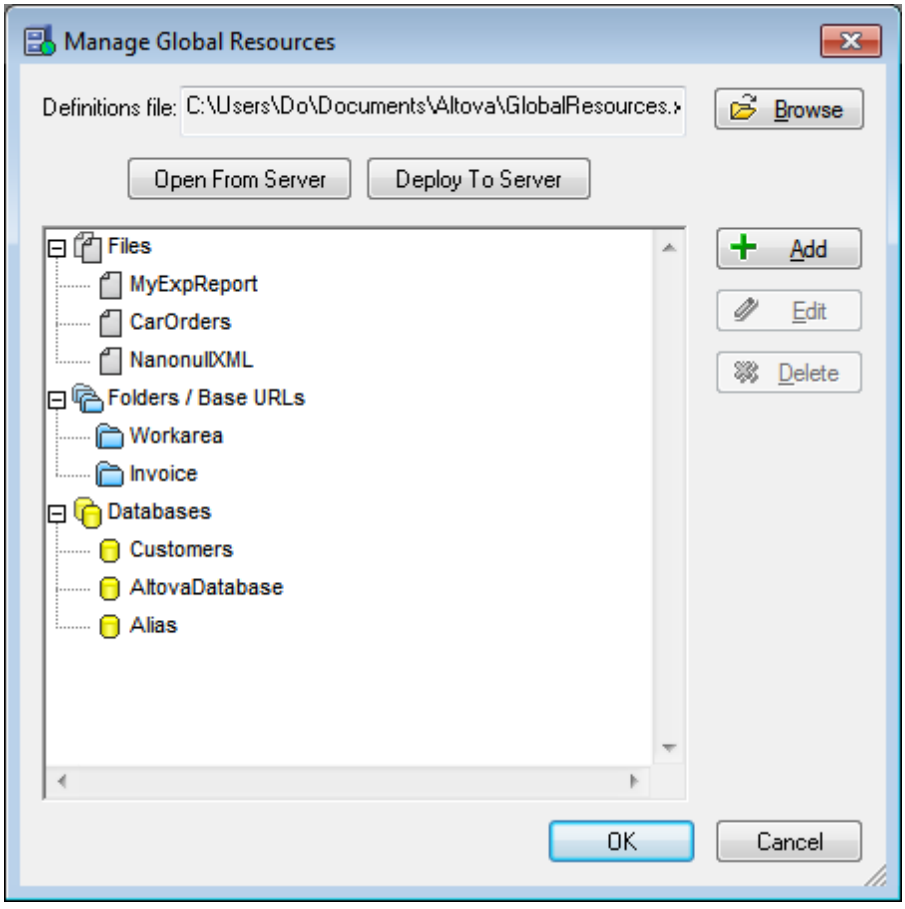
Global Resources

- 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.)
- **Open From Server** and **Deploy to Server**, respectively, enables you to open a global resource from and deploy a global resource to a MobileTogether Server.

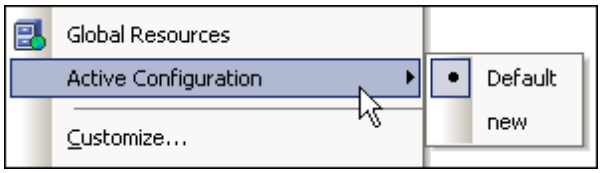


How to define global resources is described in detail in the section, Defining Global Resources.

Active Configuration

- Active Configuration
  - *Description*

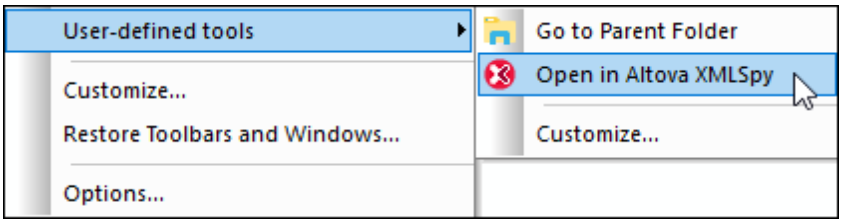
Placing the mouse over the command rolls out a submenu containing all the configurations defined in the currently active Global Resources XML File.



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.

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. 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, enabling you to go directly to the dialog in which you can edit an existing custom command or create a new custom command.

Customize

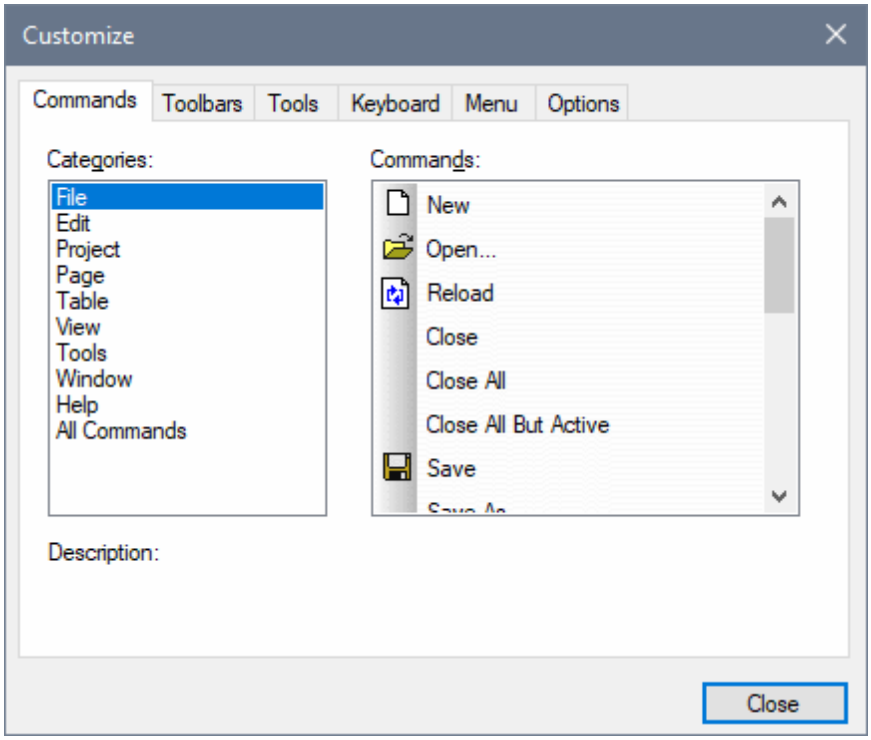
The customize command lets you customize MobileTogether Designer to suit your personal needs.

Commands
Toolbars
Tools
Keyboard
Menu
Options

Commands

▼ 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, 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** 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.

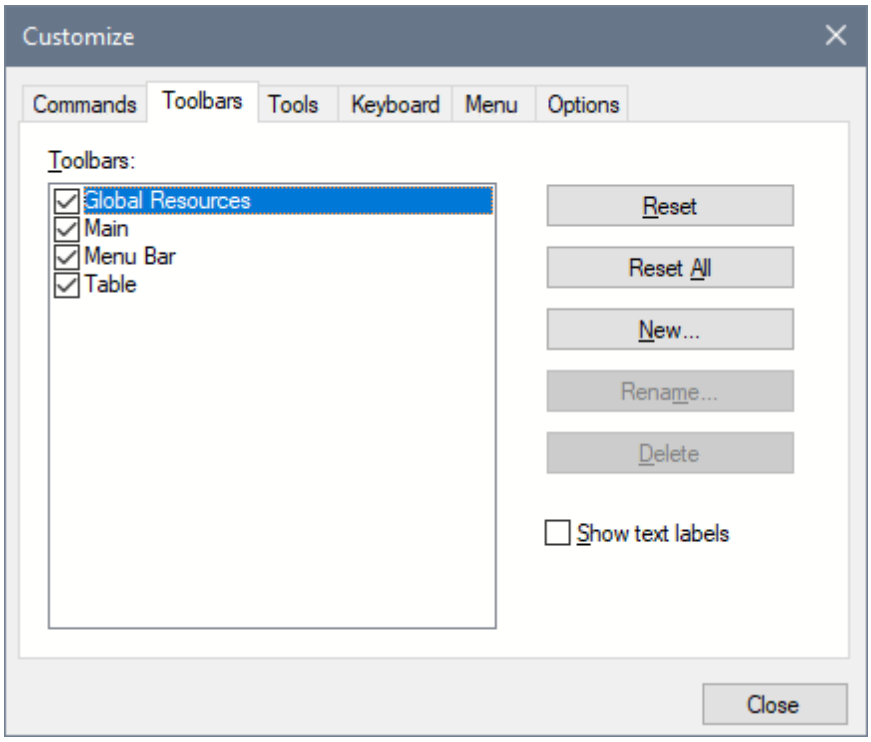
Toolbars

▼ 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 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.

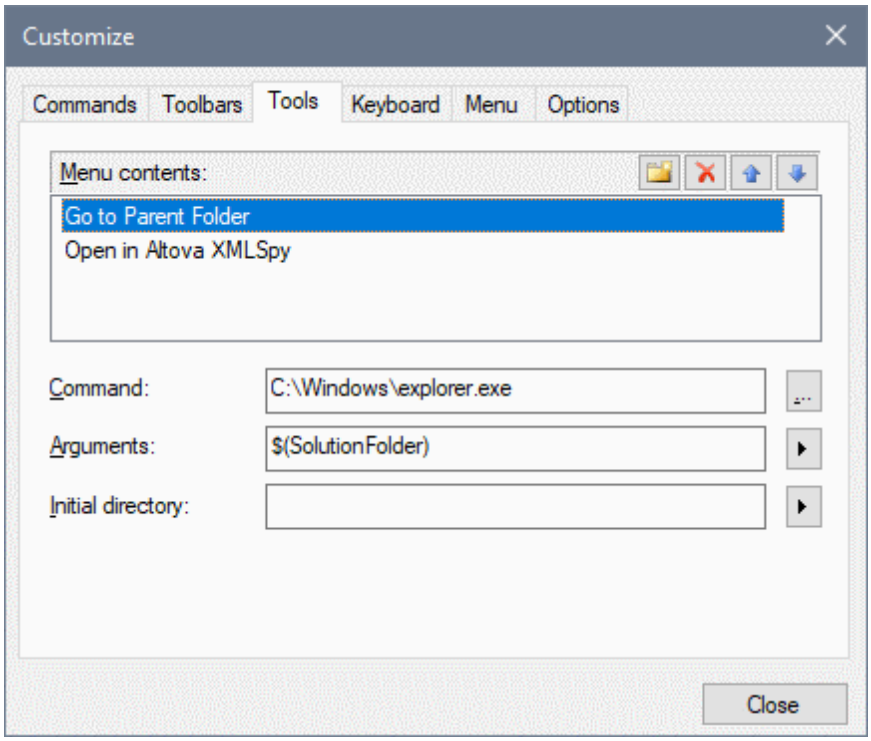
- *To delete a toolbar:* Select the toolbar you wish to delete in the Toolbars pane and click **Delete**.
- *To show text labels of commands in a particular toolbar*: Select that toolbar and click the *Show Text Labels* check box. Note that text labels have to be activated for each toolbar separately.

Note: To add a command to a toolbar, drag the command you want from the *Commands* list box in the Commands 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 for more details).

Tools

▼ 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** 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.



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** menu.

When you click a user-defined command (in the **Tools | User-defined Tools** 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.

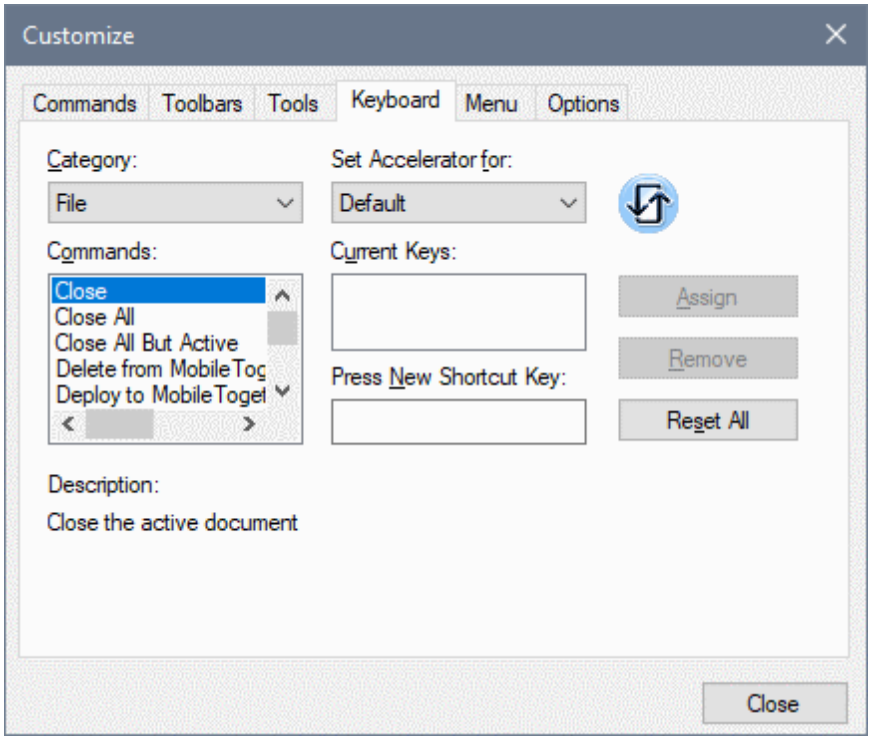
Initial directory

The *Initial Directory* entry is optional. It sets the initial directory for the command that is created.

Keyboard

▼ Keyboard

The Keyboard tab enables you to create new keyboard shortcuts for any command.



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**.

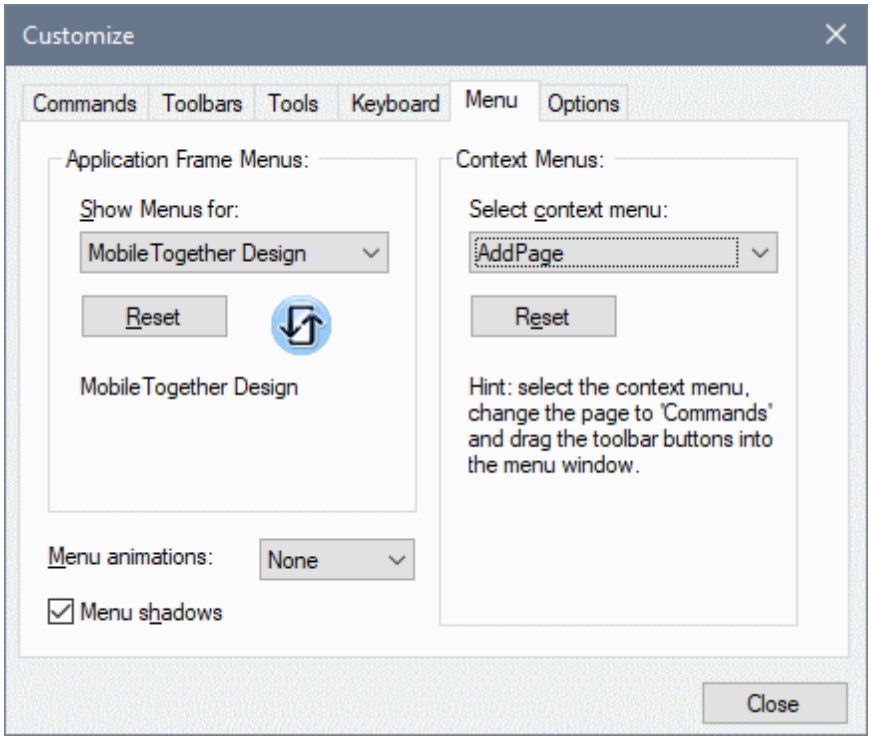
Set accelerator for

Currently no function is available.

Menu

▼ Menu

The Menu tab allows you to customize the two main menu bars (default and application menu bars) as well as context menus.



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 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.
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.

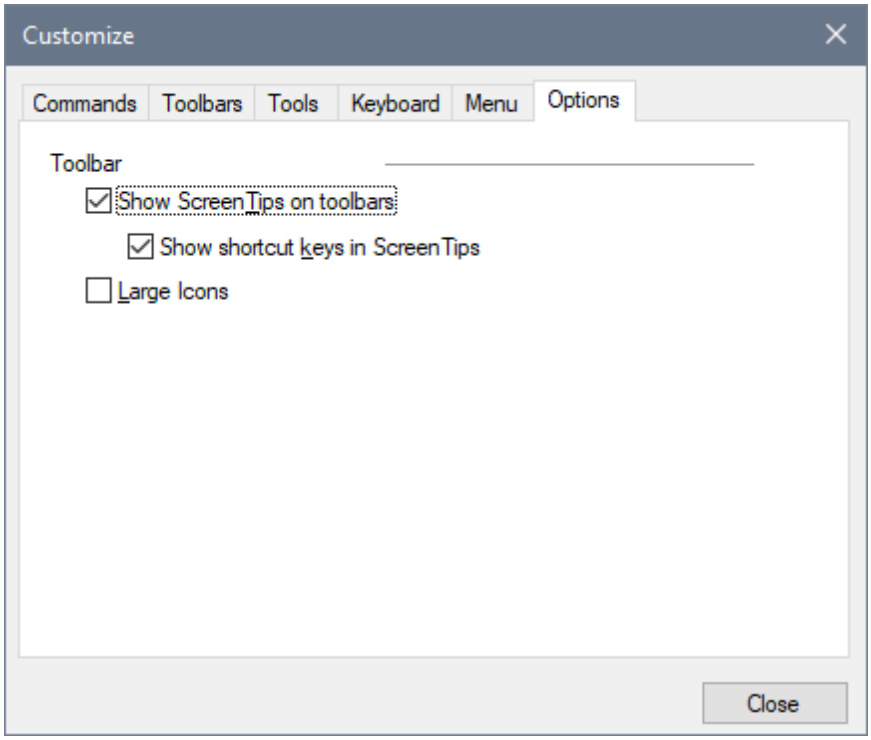
Menu shadows

Select the *Menu shadows* check box to give all menus shadows.

Options

▼ Options

The Options tab allows you to define general environment settings.



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.

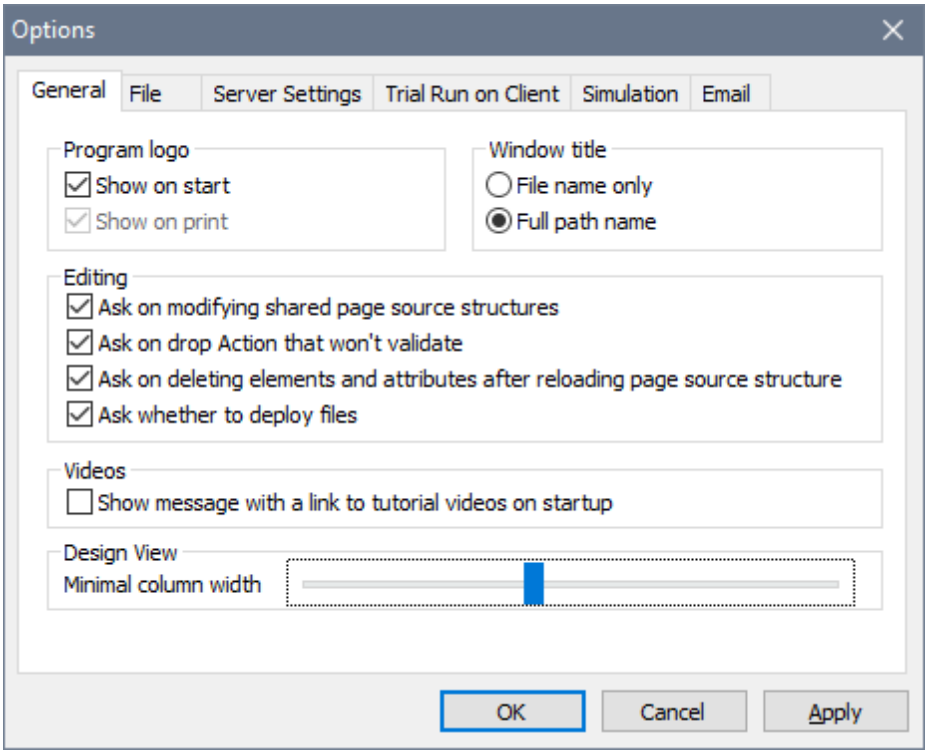
Restore Toolbars and Windows

- Restore Toolbars and Windows
  - *Description*

    Shuts down MobileTogether Designer and restarts it with all toolbars and windows reset to the original state in which they were at installation.

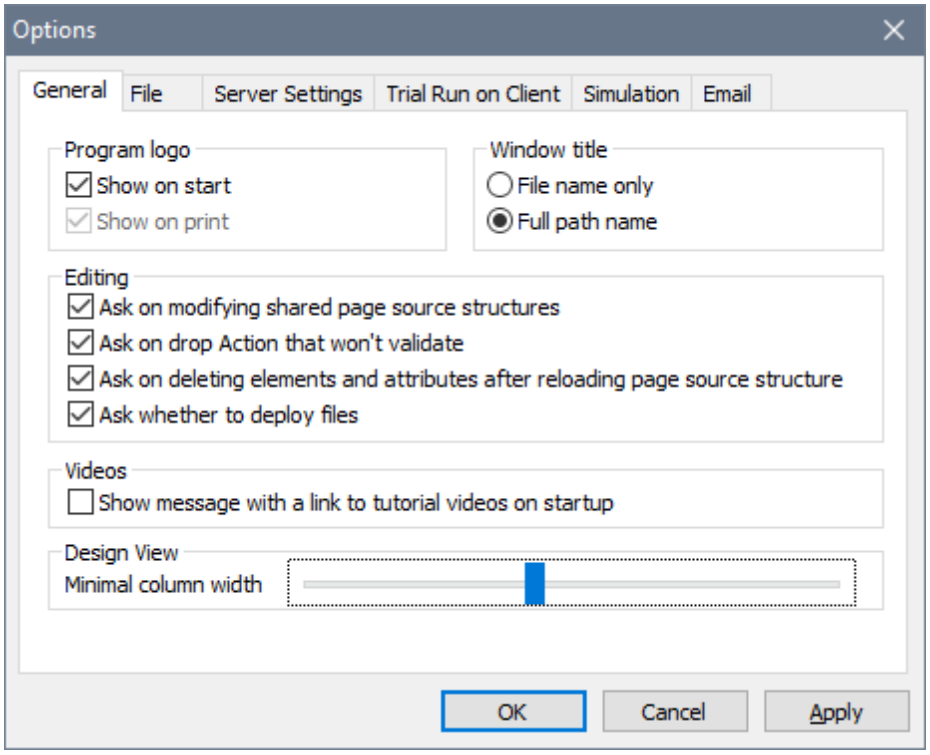
Options

The **Options** command displays the Options dialog (*screenshot below*). The settings available in the various tabs are described below.



- General

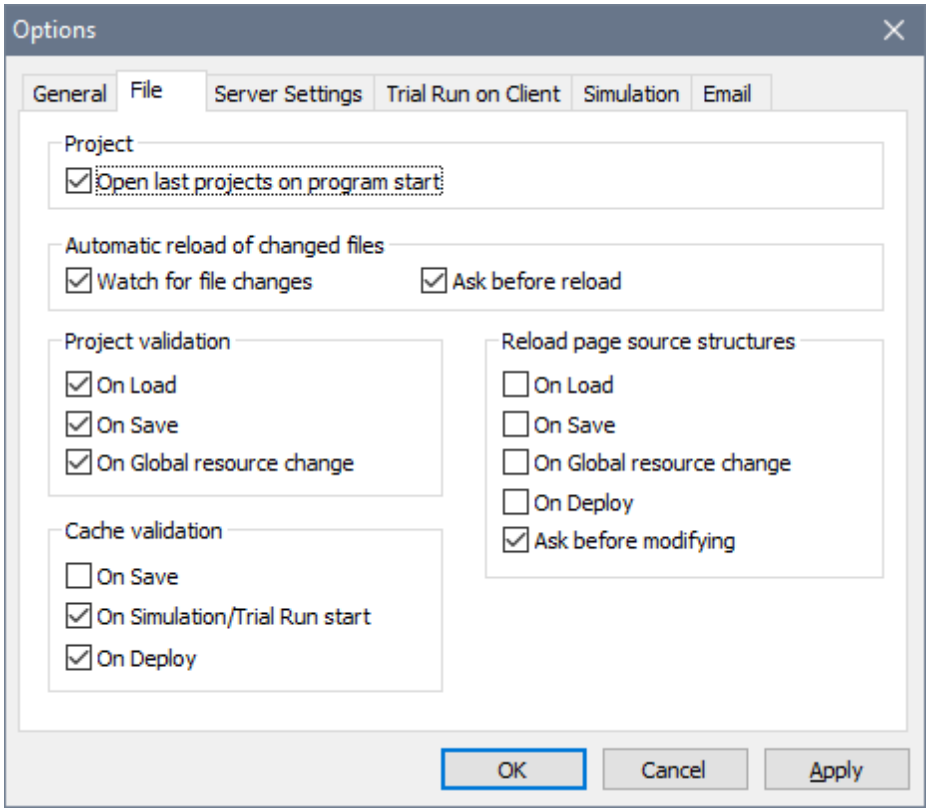
  In the General tab (*screenshot below*) you can define the settings shown in the screenshot.



- *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 | Ask:* 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.
- *Videos:* Shows 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.

▼ File

In the File tab (*screenshot below*) you can define the settings shown in the screenshot.

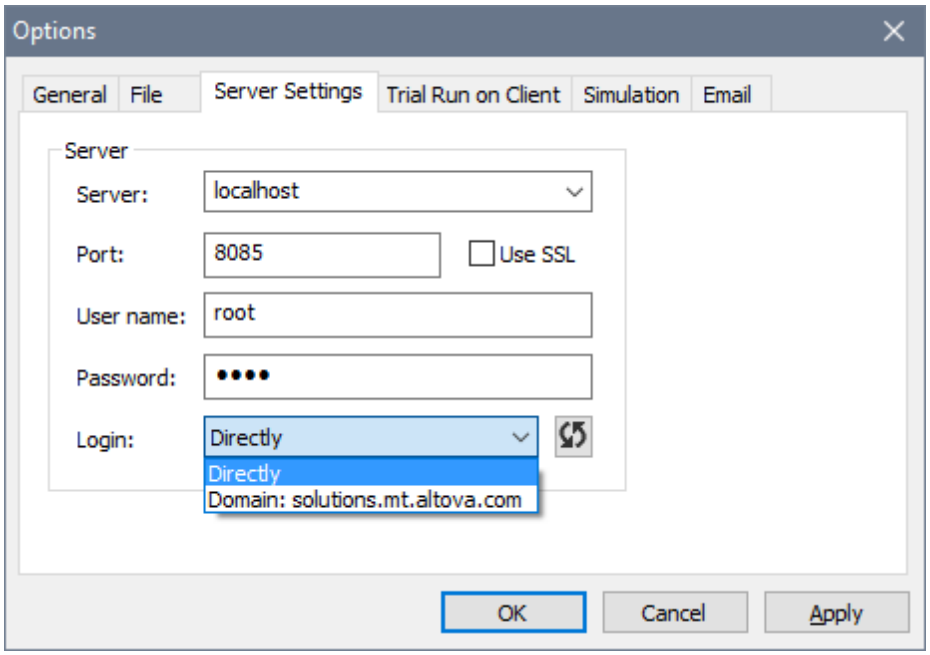


- *Project:* If selected, then on program start, all the projects are opened that were open when the program was last closed.
- *Reload of changed files:* Options to watch for changes made by another user and to ask whether to reload or not. If a file is reloaded, your own changes from the time of the 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.
- *Cache validation:* Specifies when the cache is validated. Select the options you want.

▼ 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 and when the server is used for workflow simulation. 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.

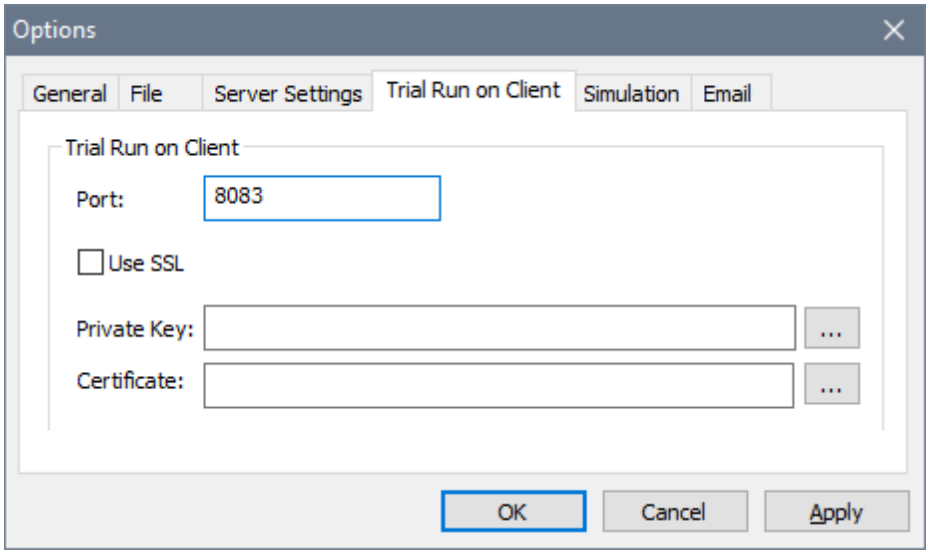


If the Active Directory login feature of MobileTogether Server has been enabled for you as a domain user, then the login information you enter for connecting MobileTogether Designer to MobileTogether Server can be your domain authentication info. For example, if your Windows user name and password on 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.

Trial Run on Client

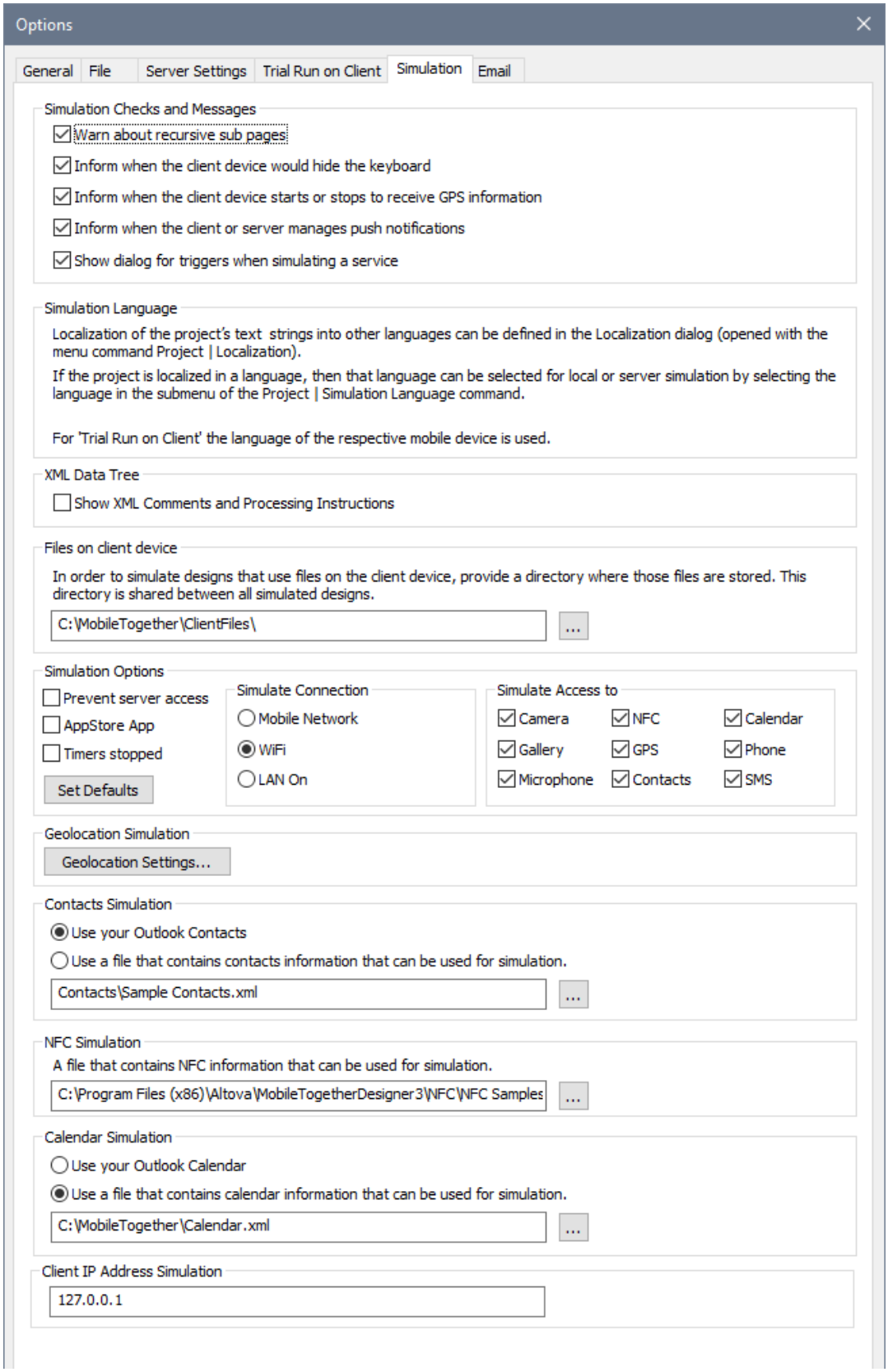
In the Trial Run on Client tab (*screenshot below*) you can define the local port via which MobileTogether Designer connects with the client. For the Trial Run on Client feature, MobileTogether Designer itself acts as the MobileTogether Server and serves the design and related data files directly to the client.



- *SSL:* Whether SSL is used.
- *Private key, certificate:* Browse for the SSL private key and certificate (if SSL is used).

Simulation

In the Simulation tab (*screenshot below*), you can specify various aspects of simulations. whether warnings should be issued about recursive sub pages and whether comment and processing instructions should be shown in the simulator's XML data tree..

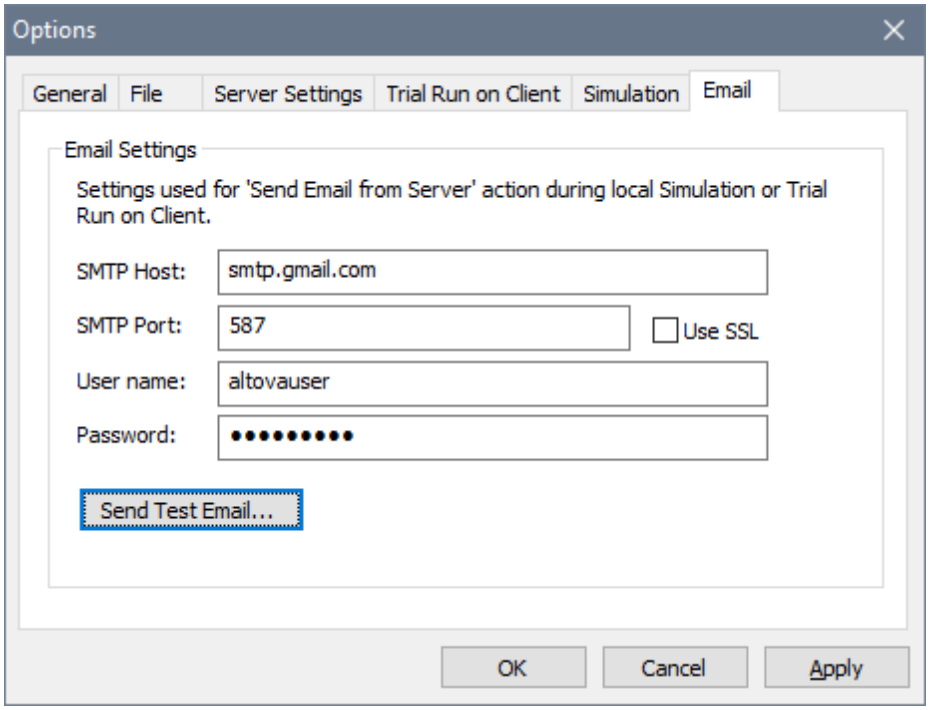

Options
General
File
Server Settings
Trial Run on Client
Simulation
Email
Simulation Checks and Messages
Warn about recursive sub pages
Inform when the client device would hide the keyboard
Inform when the client device starts or stops to receive GPS information
Inform when the client or server manages push notifications
Show dialog for triggers when simulating a service
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\
...
Simulation Options
Prevent server access
AppStore App
Timers stopped
Set Defaults
Simulate Connection
Mobile Network
WiFi
LAN On
Simulate Access to
Camera
NFC
Calendar
Gallery
GPS
Phone
Microphone
Contacts
SMS
Geolocation Simulation
Geolocation Settings...
Contacts Simulation
Use your Outlook Contacts
Use a file that contains contacts information that can be used for simulation.
Contacts\Sample Contacts.xml
...
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
...
Client IP Address Simulation
127.0.0.1

- *Simulation checks and messages:* Specifies whether warnings should be issued about (i) recursive sub pages, (ii) the keyboard being hidden on the client, (iii) client starts or stops receiving GPS information, (iv) PN management.
- *Show dialog for service triggers*: Toggles on/off the display of the dialog when a service simulation is started.
- *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**). 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** 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.
- *Options:* The settings you make here determine the default settings of the simulator. After the simulator is started, you can change these settings in the simulator's toolbar or its **Simulation** menu. You can 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, you can change the options at any time.) The **Set Defaults** button resets the options to their default settings.
- *Geolocation settings:* Enables default geolocation settings to be defined. See the section Geolocation Settings 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.
- *NFC Simulation*: This setting specifies the file to use to simulate NFC tag discovery.
- *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 a file can be used to provide calendar simulation data.
- *Client IP Address Simulation:* A random text value can be used to simulate the IP address obtained by the **`mt-client-ip-address`**.

** Microsoft Outlook is part of the Microsot Offfice suite. Among other functionality, it provides a contacts manager and a calendar.*

Email

The settings in the Email tab are used during local simulations for access to the SMTP server of an email service provider (usually your ISP). They are used by the Send Email (from Server) 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*).



- *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.

20.8 Window

The **Window** menu contains the following commands:

- Cascade
- Tile Horizontally
- Tile Vertically
- Close
- Close All
- Close All But Active
- Currently Open Windows List

Cascade and Tile

The **Window** menu has commands to specify how MobileTogether Designer windows should be displayed in the GUI (cascaded, tiled, or maximized). To maximize a window, click the **Maximize** button of that window.

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.

Currently Open Window List

All currently open document windows are listed at the bottom of the **Window** menu by document name, with the active window being checked. To make another window active, click the name of the window you wish to make active. This list shows all currently open windows, and lets you quickly switch between them. You can also use **CTRL+TAB** or **CTRL+F6** keyboard shortcuts to cycle through the open windows.

Windows dialog

At the very bottom of the list of open windows is an entry for the **Windows** dialog. Clicking this entry opens the Windows dialog, which displays a list of all open windows and provides commands that can be applied to the selected window/s. (A window is selected by clicking on its name.)

Warning: To exit the Windows dialog, click **OK**. Do **not** click the **Close Window(s)** button. The **Close Window(s)** button closes the window/s currently selected in the Windows dialog.

20.9 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 also contains the Registration dialog, which lets you enter your license key-code once you have purchased the product.

The **Help** menu contains the following commands:

- Table of Contents
- Index
- Search
- Software Activation
- Order Form
- Registration
- Check for Updates
- Support Center
- Show Video Demos
- MobileTogether Designer on the Internet
- About MobileTogether Designer

Table of Contents, Index, Search

- Table of Contents
  - *Description*

    Opens the onscreen help manual of MobileTogether Designer with the Table of Contents displayed in the left-hand-side pane of the Help window. The Table of Contents provides an overview of the entire Help document. Clicking an entry in the Table of Contents takes you to that topic.

- Index
  - *Description*

    Opens the onscreen help manual of MobileTogether Designer with the Keyword Index displayed in the left-hand-side pane of the Help window. The index lists keywords and lets you navigate to a topic by double-clicking the keyword. If a keyword is linked to more than one topic, a list of these topics is displayed.

- Search
  - *Description*

    Opens the onscreen help manual of MobileTogether Designer with the Search dialog displayed in the left-hand-side pane of the Help window. To search for a term, enter the term in the input field, and press **Return**. The Help system performs a full-text search on the entire Help documentation and returns a list of hits. Double-click any item to display that item.

Activation, Order Form, Registration, Updates

- Software Activation
  - *Description*

    After you download your Altova product software, you can activate it using either a free evaluation key or a purchased permanent license key.

    - ***Free evaluation key.*** 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 evaluation key-code. Enter your name, company, and e-mail address in the dialog that appears, and click Request Now! The evaluation key is sent to the e-mail address you entered and should reach you in a few minutes. Now enter the key in the key-code field of the Software Activation dialog box and click **OK** to start working with your Altova product. The software will be unlocked for a period of 30 days.
    - ***Permanent license key.*** The Software Activation dialog contains a button to purchase a permanent license key. Clicking this button takes you to Altova's online shop, where you can purchase a permanent license key for your product. There are two types of permanent license: single-user and multi-user. Both will be sent to you by e-mail. A *single-user license* contains your license-data and includes your name, company, e-mail, and key-code. A *multi-user license* contains your license-data and includes your company name and key-code. Note that your license agreement does not allow you to install more than the licensed number of copies of your Altova software on the computers in your organization (per-seat license). Please make sure that you enter the data required in the registration dialog exactly as given in your license e-mail.

    Note: When you enter your license information in the Software Activation dialog, ensure that you enter the data exactly as given in your license e-mail. For multi-user licenses, each user should enter his or her own name in the Name field.

    The Software Activation dialog can be accessed at any time by clicking the **Help | Software Activation** command.

- Order Form
  - *Description*

    When you are ready to order a licensed version of the software product, you can use either the **Order license key** button in the Software Activation dialog (*see previous section*) or the **Help | Order Form** command to proceed to the secure Altova Online Shop.

- Registration
  - *Description*

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
  - *Description*

    Checks with the Altova server whether a newer version than yours is currently available and displays a message accordingly.

Other Commands

- Support Center
  - *Description*

    A link to the Altova Support Center on the Internet. The Support Center provides FAQs, discussion forums where problems are discussed, and access to Altova's technical support staff.

- 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.

Chapter 21

Frequently Asked Questions

21 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 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+

Chapter 22

Appendices

22 Appendices

Information included in this section:

22.1 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.

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 XSLT and XQuery engines read non-prefixed functions as belonging to a default functions namespace which is that specified in the XPath/XQuery functions specifications`http://www.w3.org/2005/xpath-functions`. 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 timezone 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.

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

XPath/XQuery functions

XPath/XQuery functions can be used both in XPath expressions as well as in XQuery expressions:

- Date/Time
- Geolocation
- Image-related
- Numeric
- Sequence
- String

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
- Add a duration to xs:date and return xs:date
- Add a duration to xs:time and return xs:time
- Format and retrieve durations
- Remove timezone from functions that generate current date/time
- Return weekday as integer from date
- Return week number as integer from date
- Build date, time, or duration type from lexical components of each type
- Construct date, dateTime, or time type from string input
- Age-related functions

▼ Grouped alphabetically

altova:add-days-to-date
altova:add-days-to-dateTime
altova:add-hours-to-dateTime
altova:add-hours-to-time
altova:add-minutes-to-dateTime
altova:add-minutes-to-time
altova:add-months-to-date
altova:add-months-to-dateTime
altova:add-seconds-to-dateTime
altova:add-seconds-to-time
altova:add-years-to-date
altova:add-years-to-dateTime
altova:age
altova:age-details
altova:build-date

[Top]

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]

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]

Format and retrieve durations 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.

▼ 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]

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]

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-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-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-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

[Top]

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. The weekdays are numbered starting with `Monday=1`. 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. The weekdays are numbered starting with `Monday=1`. 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]

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]

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 arguments 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]

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]

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]

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

▼ 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` function and the Exif metadata's `@Geolocation` 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`, `W/E`)
  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/W**) 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`, `W/E`)
  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/W**) is optional
  +/-D°M.MM' +/-D°M.MM'
  Example: **+33°55.55' -22°44.44'**

- Decimal degrees, with suffixed orientation (`N/S`, `W/E`)
  D.DDN/S D.DDW/E
  Example: **33.33N 22.22W**

- Decimal degrees, with prefixed sign (`+/-`); the plus sign for (**N/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 function and the Exif metadata's @Geolocation 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, W/E)
  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/W**) 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, W/E)
  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/W**) is optional
  +/-D°M.MM' +/-D°M.MM'
  Example: **+33°55.55' -22°44.44'**

- Decimal degrees, with suffixed orientation (N/S, W/E)
  D.DDN/S D.DDW/E
  Example: **33.33N 22.22W**

- Decimal degrees, with prefixed sign (+/-); the plus sign for (**N/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 function and the Exif metadata's @Geolocation 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, W/E)
  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/W**) 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, W/E)
  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/W**) is optional
  +/-D°M.MM' +/-D°M.MM'
  Example: **+33°55.55' -22°44.44'**

- Decimal degrees, with suffixed orientation (N/S, W/E)
  D.DDN/S D.DDW/E
  Example: **33.33N 22.22W**

- Decimal degrees, with prefixed sign (+/-); the plus sign for (**N/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-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 function and the Exif metadata's @Geolocation 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, W/E)
    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/W**) 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, W/E)
    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/W**) is optional

+/-D°M.MM' +/-D°M.MM'
Example: **+33°55.55' -22°44.44'**

- Decimal degrees, with suffixed orientation (N/S, W/E)
  D.DDN/S D.DDW/E
  Example: **33.33N 22.22W**

- Decimal degrees, with prefixed sign (+/-); the plus sign for (**N/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 function and the Exif metadata's @Geolocation 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, W/E)
  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/W**) 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, W/E)
  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/W**) is optional
  +/-D°M.MM' +/-D°M.MM'
  Example: **+33°55.55' -22°44.44'**

- Decimal degrees, with suffixed orientation (N/S, W/E)
  D.DDN/S D.DDW/E
  Example: **33.33N 22.22W**

- Decimal degrees, with prefixed sign (+/-); the plus sign for (**N/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]

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

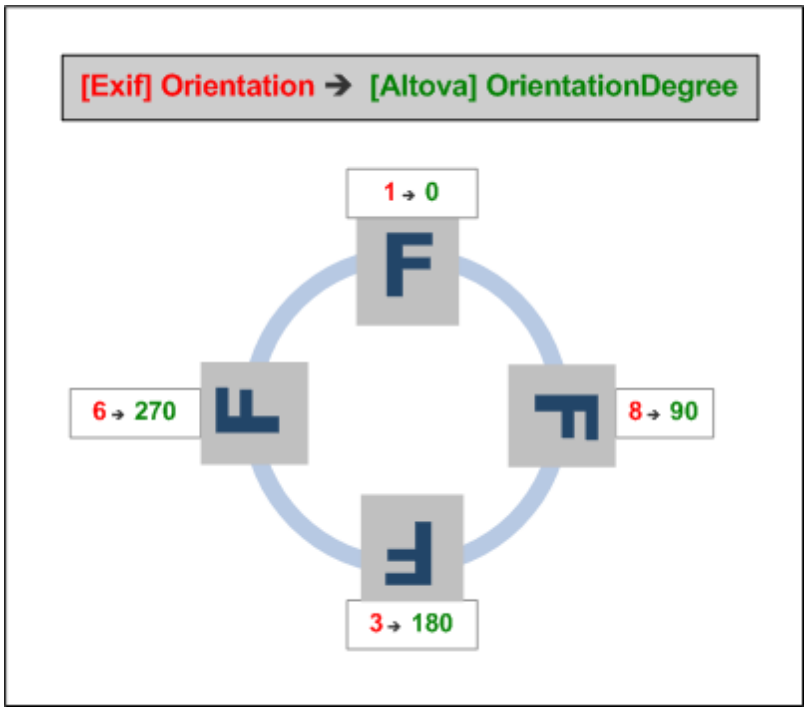
- **`mt-transform-image`**`(Images/Image[@id='43'], '50%', 90, 75)`
  The function takes as its input an image that is stored as a Base64-encoded string in the descendant `Images/Image` node that has an `@id` value of `43`. The function returns a transformed image. The transformed image is resized to 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

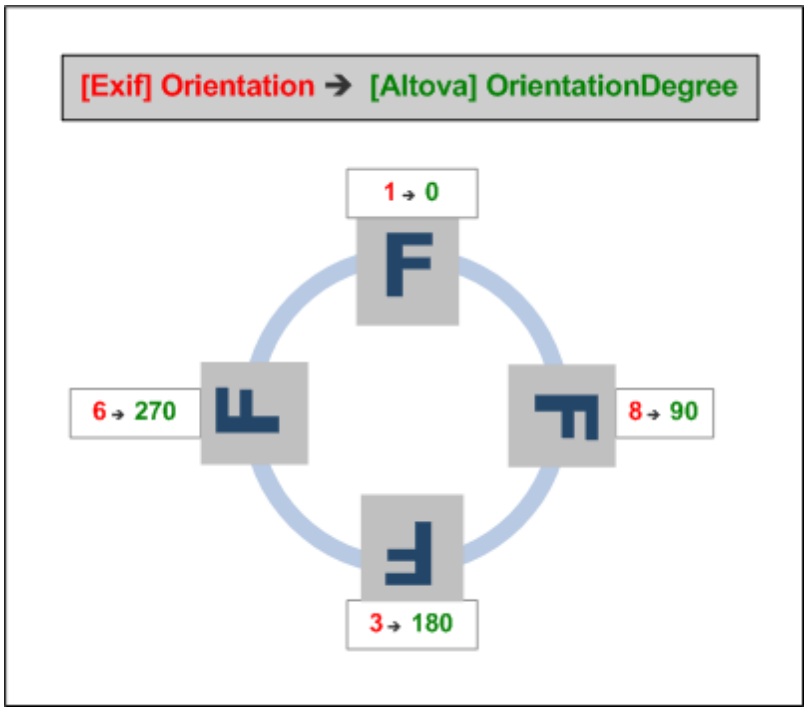
The Altova XPath/XQuery Engine generates the custom attribute **`Geolocation`** from standard Exif metadata tags. **`Geolocation`** is a concatenation of four Exif tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, with units added (*see table below*).

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]

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`.

[Top]

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'

[Top]

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

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")`.

[Top]

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()*
- **attributes**("MyElem", "rip") returns MyElement()*
- **attributes**("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.*

▼ 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)`

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.

▼ 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

▼ 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"

XPath/XQuery Functions: Miscellaneous

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

▼ 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`

22.2 License Information

This section contains:

- Information about the distribution of this software product
- Information about software activation and license metering
- Information about the intellectual property rights related to this software product
- The End-User License Agreement governing the use of this software product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

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 before making a purchasing decision.
- Once you decide to buy the software, you can place your order online at the Altova website and immediately 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 a comprehensive integrated onscreen help system. The latest version of the user manual is available at www.altova.com (i) in HTML format for online browsing, and (ii) in 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 only distribute the Setup programs, provided that they are not modified in any way. Any person that 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.

For further details, please refer to the Altova Software License Agreement at the end of this section.

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 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 website (http://www.iana.org/) *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.

You will also notice that, if you are online, your Altova product contains many useful functions; these are unrelated to the license-metering technology.

Intellectual Property Rights

The Altova Software and any copies that you are authorized by Altova to make are the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

Altova software contains certain Third Party Software that is also protected by intellectual property laws, including without limitation applicable copyright laws as described in detail at http://www.altova.com/legal_3rdparty.html.

All other names or trademarks are the property of their respective owners.

Altova MobileTogether Designer End User License Agreement

THIS IS A LEGAL DOCUMENT -- RETAIN FOR YOUR RECORDS

ALTOVA® MOBILETOGETHER DESIGNER END USER LICENSE AGREEMENT

Licensor:
Altova GmbH
Rudolfsplatz 13a/9
A-1010 Wien
Austria

Important - Read Carefully. Notice to User:

This End User License Agreement ("Agreement") is a legal document between you and Altova GmbH ("Altova"). It is important that you read this document before using the Altova-provided software ("Software") and any accompanying documentation, including, without limitation printed materials, 'online' files, or electronic documentation ("Documentation"). By clicking the "I accept" and "Next" buttons below, or by installing, or otherwise using the Software, you agree to be bound by the terms of this Agreement as well as the Altova Privacy Policy ("Privacy Policy") including, without limitation, the warranty disclaimers, limitation of liability, data use and termination provisions below, whether or not you decide to purchase the Software. You agree that this agreement is enforceable like any written agreement negotiated and signed by you. If you do not agree, you are not licensed to use the Software, and you must destroy any downloaded copies of the Software in your possession or control. You may print a copy of this Agreement as part of the installation process at the time of acceptance. Alternatively, a copy of this Agreement may be found at http://www.altova.com/legal.html and a copy of the Privacy Policy may be found at http://www.altova.com/privacy.

1. SOFTWARE LICENSE

(a) License Grant.

(i) Upon your acceptance of this Agreement Altova grants you a non-exclusive, non-transferable (except as provided below), limited license, without the right to grant sublicenses, to install and use a copy of the Software on one compatible personal computer or workstation.

(ii) You may not use the Software to develop and distribute other software programs that directly compete with any Altova software or service without prior written permission. Altova reserves all other rights in and to the Software.

(b) AppStore Apps.

(i) The AppStore App feature in the Software enables users to generate source code. Your license to install and use a copy of the Software as provided herein permits you to generate source code based on (i) Altova Library modules that are included in the Software (such generated code hereinafter referred to as the "Restricted Source Code") and (ii) schemas or mappings that you create or provide (such code as may be generated from your schema or mapping source materials hereinafter referred to as the "Unrestricted Source Code"). In addition to the rights granted herein, Altova grants you a non-exclusive, non-transferable, limited license to compile the complete generated code (comprised of the combination of the Restricted Source Code and the Unrestricted Source Code) into executable object code form, and to use, copy, distribute or license that executable. You may not distribute or redistribute, sublicense, sell, or

transfer the Restricted Source Code to a third-party in the un-compiled form. Notwithstanding anything to the contrary herein, you may not distribute, incorporate or combine with other software, or otherwise use the Altova Library modules or Restricted Source Code, or any Altova intellectual property embodied in or associated with the Altova Library modules or Restricted Source Code, in any manner that would subject the Restricted Source Code to the terms of a copyleft, free software or open source license that would require the Restricted Source Code or Altova Library modules source code to be disclosed in source code form. Notwithstanding anything to the contrary herein, you may not use the Software to develop and distribute other software programs that directly compete with any Altova software or service without prior written permission. Altova reserves all other rights in and to the Software.

(ii) In the event Restricted Source Code is incorporated into executable object code form, you will include the following statement in (1) introductory splash screens, or if none, within one or more screens readily accessible by the end-user, and (2) in the electronic and/or hard copy documentation: "Portions of this program were developed using Altova® MobileTogether and includes libraries owned by Altova GmbH, Copyright © 2007-2016 Altova GmbH (www.altova.com)."

(iii) You agree not to use the Software to create an AppStore App for any unlawful or illegal activity, nor to develop any AppStore App that would commit or facilitate the commission of a crime, or other tortious, unlawful, or illegal act. You agree that, to the best of your knowledge and belief, your AppStore App will not violate, misappropriate, or infringe any Altova or third party copyrights, trademarks, rights of privacy and publicity, trade secrets, patents, or other proprietary or legal rights (e.g. musical composition or performance rights, video rights, photography or image rights, logo rights, third party data rights, etc. for content and materials that may be included in your AppStore App). Further, you agree not to use the Software to create any AppStore App or other software program that would disable, hack or otherwise interfere with any security, digital signing, digital rights management, content protection, verification or authentication mechanisms implemented in or by the Software or by other third party software, services or technology, or enable others to do so, unless otherwise permitted by the Altova in writing. AppStore Apps must not contain any malware, malicious or harmful code, program, or other internal component (e.g. computer viruses, trojan horses, "backdoors") and may not use any Software in a way that could damage, destroy, or adversely affect Altova software or services, or any other software, firmware, hardware, data, systems, services, or networks.

(iv) You agree that you are solely responsible for any data, content, or resources that you create, transmit or display through the AppStore App, and for the consequences of your actions.

(c) Backup and Archival Copies. You may make one (1) backup and one (1) archival copy of the Software, provided your backup and archival copies are not installed or used on any computer and further provided that all such copies shall bear the original and unmodified copyright, patent and other intellectual property markings that appear on or in the Software.

(d) Title. Title to the Software is not transferred to you. Ownership of all copies of the Software and of copies made by you is vested in Altova, subject to the rights of use granted to you in this Agreement. As between you and Altova, documents, files, stylesheets, generated program code (including the Unrestricted Source Code), schemas that are authored or created by you via your utilization of the Software, and AppStore Apps, in accordance with its Documentation and the terms of this Agreement, are your property. "AppStore Apps" means the app authored or created by you via your utilization of the "create AppStore App" option in Altova MobileTogether Designer and connected to the Altova MobileTogether Server.

(e) Reverse Engineering. Except and to the limited extent as may be otherwise specifically provided by applicable law in the European Union, you may not reverse engineer, decompile, disassemble or otherwise attempt to discover the source code, underlying ideas, underlying user interface techniques or algorithms of the Software by any means whatsoever, directly or indirectly, or disclose any of the foregoing, except to the extent you may be expressly permitted to decompile under applicable law in the European Union, if it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested Altova to provide the information necessary to achieve such operability and Altova has not made such information available. Altova has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by Altova or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software. Requests for information from users in the European Union with respect to the above should be directed to the Altova Customer Support Department.

(f) Other Restrictions. You may not loan, rent, lease, sublicense, distribute or otherwise transfer all or any portion of the Software to third parties except as otherwise expressly provided. You may not copy the Software except as expressly set forth herein, and any copies that you are permitted to make pursuant to this Agreement must contain the same copyright, patent and other intellectual property markings that appear on or in the Software. You may not modify, adapt or translate the Software. You may not, directly or indirectly, encumber or suffer to exist any lien or security interest on the Software; knowingly take any action that would cause the Software to be placed in the public domain; or use the Software in any computer environment not specified in this Agreement. You may not permit any use of or access to the Software by any third party in connection with a commercial service offering, such as for a cloud-based or web-based SaaS offering.

You will comply with applicable law and Altova's instructions regarding the use of the Software. You agree to notify your employees and agents who may have access to the Software of the restrictions contained in this Agreement and to ensure their compliance with these restrictions.

(g) NO GUARANTEE. THE SOFTWARE IS NEITHER GUARANTEED NOR WARRANTED TO BE ERROR-FREE NOR SHALL ANY LIABILITY BE ASSUMED BY ALTOVA IN THIS RESPECT. NOTWITHSTANDING ANY SUPPORT FOR ANY TECHNICAL STANDARD, THE SOFTWARE IS NOT INTENDED FOR USE IN OR IN CONNECTION WITH, WITHOUT LIMITATION, THE OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION, COMMUNICATION SYSTEMS, AIR TRAFFIC CONTROL EQUIPMENT, MEDICAL DEVICES OR LIFE SUPPORT SYSTEMS, MEDICAL OR HEALTH CARE APPLICATIONS, OR OTHER APPLICATIONS WHERE THE FAILURE OF THE SOFTWARE OR ERRORS IN DATA PROCESSING COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR THE ACCURACY AND ADEQUACY OF THE SOFTWARE AND ANY DATA GENERATED OR PROCESSED BY THE SOFTWARE FOR YOUR INTENDED USE AND YOU WILL DEFEND, INDEMNIFY AND HOLD ALTOVA, ITS OFFICERS AND EMPLOYEES HARMLESS FROM ANY THIRD PARTY CLAIMS, DEMANDS, OR SUITS THAT ARE BASED UPON THE ACCURACY AND ADEQUACY OF THE SOFTWARE IN YOUR USE OR ANY DATA GENERATED BY THE SOFTWARE IN YOUR USE.

2. INTELLECTUAL PROPERTY RIGHTS

You acknowledge that the Software and any copies that you are authorized by Altova to make are

the intellectual property of and are owned by Altova and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of Altova and its suppliers. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You acknowledge that Altova retains the ownership of all patents, copyrights, trade secrets, trademarks and other intellectual property rights pertaining to the Software, and that Altova's ownership rights extend to any images, photographs, animations, videos, audio, music, text and "applets" incorporated into the Software and all accompanying printed materials. You will take no actions which adversely affect Altova's intellectual property rights in the Software. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademark owners' names. Trademarks may only be used to identify printed output produced by the Software, and such use of any trademark does not give you any right of ownership in that trademark. Altova®, XMLSpy®, Authentic®, StyleVision®, MapForce®, UModel®, DatabaseSpy®, DiffDog®, SchemaAgent®, SemanticWorks®, MissionKit®, Markup Your Mind®, Nanonull™, MobileTogether®, MobileTogether Server®, MobileTogether Designer®, MobileTogether Mobile App®, RaptorXML®, RaptorXML Server®, RaptorXML +XBRL Server®, Powered By RaptorXML®, FlowForce Server®, StyleVision Server®, and MapForce Server® are trademarks of Altova GmbH. (pending or registered in numerous countries).Unicode and the Unicode Logo are trademarks of Unicode, Inc. Windows, Windows XP, Windows Vista, Windows 7, and Windows 8 are trademarks of Microsoft. W3C, CSS, DOM, MathML, RDF, XHTML, XML and XSL are trademarks (registered in numerous countries) of the World Wide Web Consortium (W3C); marks of the W3C are registered and held by its host institutions, MIT, INRIA and Keio. Except as expressly stated above, this Agreement does not grant you any intellectual property rights in the Software. Notifications of claimed copyright infringement should be sent to Altova's copyright agent as further provided on the Altova Web Site.

3. LIMITED WARRANTY AND LIMITATION OF LIABILITY

(a) Limited Warranty and Customer Remedies. YOU ACKNOWLEDGE THAT THE SOFTWARE IS PROVIDED TO YOU **"AS-IS" WITH NO WARRANTIES FOR USE OR PERFORMANCE,** AND ALTOVA DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, WHETHER EXPRESS OR IMPLIED. WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR THE SOFTWARE, BUT IT MAY BE LIMITED, ALTOVA'S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (USD $50) IN TOTAL.

(b) No Other Warranties and Disclaimer. THE FOREGOING LIMITED WARRANTY AND REMEDIES STATE THE SOLE AND EXCLUSIVE REMEDIES FOR ALTOVA OR ITS SUPPLIER'S BREACH OF WARRANTY. ALTOVA AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, ALTOVA AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, ALTOVA AND ITS SUPPLIERS DISCLAIM ALL OTHER WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, INFORMATIONAL CONTENT OR ACCURACY, QUIET ENJOYMENT, TITLE AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS LIMITED WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE

OTHERS, WHICH VARY FROM STATE/JURISDICTION TO STATE/JURISDICTION.

(c) Limitation of Liability. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW EVEN IF A REMEDY FAILS ITS ESSENTIAL PURPOSE, IN NO EVENT SHALL ALTOVA OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF ALTOVA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Because some states and jurisdictions do not allow the exclusion or limitation of liability, the above limitation may not apply to you. In such states and jurisdictions, Altova's liability shall be limited to the greatest extent permitted by law and the limitations or exclusions of warranties and liability contained herein do not prejudice applicable statutory consumer rights of person acquiring goods otherwise than in the course of business. The disclaimer and limited liability above are fundamental to this Agreement between Altova and you.

(d) Waiver and Indemnity. BY USING THE SOFTWARE, YOU AGREE, TO THE EXTENT PERMITTED BY LAW, TO SAVE AND PROTECT, HOLD HARMLESS, INDEMNIFY AND DEFEND ALTOVA, ITS DIRECTORS, OFFICERS, EMPLOYEES, AFFILIATES, AGENTS, CONTRACTORS, AND LICENSORS AGAINST ANY AND ALL LIABILITY, CAUSES OF ACTION, CLAIMS, LOSS DAMAGE OR COST AND EXPENSE ARISING FROM, ALLEGEDLY ARISING FROM, OR RESULTING DIRECTLY OR INDIRECTLY FROM ANY ACTS OF THE LICENSEE OR ANY OF ITS OFFICERS, EMPLOYEES, INDEPENDENT CONTRACTORS OR AGENTS DONE IN THE PERFORMANCE, OPERATION, OR USE OF THE SOFTWARE, OR ANY ACT DONE UNDER PRETENDED AUTHORITY OF THIS LICENSE. THIS AGREEMENT TO INDEMNIFY AND HOLD ALTOVA HARMLESS SHALL INCLUDE ANY COSTS INCURRED BY ALTOVA IN DEFENDING ANY ACTION INVOLVING AN ACT BY THE LICENSEE OR ANY OF ITS OFFICERS, EMPLOYEES, INDEPENDENT CONTRACTORS OR AGENTS, AND SHALL INCLUDE ANY ATTORNEY'S FEES INCURRED BY ALTOVA.

4. SUPPORT AND MAINTENANCE

Altova offers "Support & Maintenance Package(s)" ("SMP") for the Software only if you have obtained a valid license for MobileTogether Server, which may be obtained from Altova at www.altova.com. The Support Period shall coincide with the MobileTogether Server Software license term. The terms of SMP are set forth in the Altova MobileTogether Server Software License Agreement located at http://www.altova.com/license_agreements.html.

5. SOFTWARE ACTIVATION AND UPDATES

(a) License Metering. The Software includes a built-in license metering module that is designed to assist you with monitoring license compliance in small local networks. The metering module attempts to communicate with other machines on your local area network. You permit Altova to use your internal network for license monitoring for this purpose. This license metering module may be used to assist with your license compliance but should not be the sole method. Should your firewall settings block said communications, you must deploy an accurate means of monitoring usage by the end user and preventing users from using the Software more than the Permitted Number.

(b) License Compliance Monitoring. You are required to utilize a process or tool to ensure that the Permitted Number is not exceeded. Without prejudice or waiver of any potential violations of the Agreement, Altova may provide you with additional compliance tools should you

be unable to accurately account for license usage within your organization. If provided with such a tool by Altova, you (a) are required to use it in order to comply with the terms of this Agreement and (b) permit Altova to use your internal network for license monitoring and metering and to generate compliance reports that are communicated to Altova from time to time.

(c) 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 between your computer and the Altova license server. You agree that Altova may use these measures and you agree to follow any applicable requirements. You further agree that use of license key codes that are not or were not generated by Altova and lawfully obtained from Altova, or an authorized reseller as part of an effort to activate or use the Software violates Altova's intellectual property rights as well as the terms of this Agreement. You agree that efforts to circumvent or disable Altova's copyright protection mechanisms or License Server violate Altova's intellectual property rights as well as the terms of this Agreement. Altova expressly reserves the rights to seek all available legal and equitable remedies to prevent such actions and to recover lost profits, damages and costs.**

(d) LiveUpdate. Altova provides a new LiveUpdate notification service to you, which is free of charge. Altova may use your internal network and Internet connection for the purpose of transmitting license-related data to an Altova-operated LiveUpdate server to validate your license at appropriate intervals and determine if there is any update available for you.

(e) Use of Data. The terms and conditions of the Privacy Policy are set out in full at http://www.altova.com/privacy and are incorporated by reference into this Agreement. By your acceptance of the terms of this Agreement and/or use of the Software, you authorize the collection, use and disclosure of information collected by Altova for the purposes provided for in this Agreement and/or the Privacy Policy. Altova has the right in its sole discretion to amend this provision of the Agreement and/or Privacy Policy at any time. You are encouraged to review the terms of the Privacy Policy as posted on the Altova Web site from time to time.

(f) Audit Rights. You agree that Altova may audit your use of the Software for compliance with the terms of this Agreement at any time, upon reasonable notice. In the event that such audit reveals any use of the Software by you other than in full compliance with the terms of this Agreement, you shall reimburse Altova for all reasonable expenses related to such audit in addition to any other liabilities you may incur as a result of such non-compliance.

(g) Notice to European Users. Please note that the information as described in paragraph 5(d) above may be transferred outside of the European Economic Area, for purposes of processing, analysis, and review, by Altova, Inc., a company located in Beverly, Massachusetts, U.S.A., or its subsidiaries or Altova's subsidiaries or divisions, or authorized partners, located worldwide. You are advised that the United States uses a sectoral model of privacy protection that relies on a mix of legislation, governmental regulation, and self-regulation. You are further advised that the Council of the European Union has found that this model does not provide "adequate" privacy protections as contemplated by Article 25 of the European Union's Data Directive. (Directive 95/46/EC, 1995 O.J. (L 281) 31). Article 26 of the European Union's Data Directive allows for transfer of personal data from the European Union to a third country if the individual has unambiguously given his consent to the transfer of personal information, regardless of the third country's level of protection. By agreeing to this Agreement, you consent to the transfer of all such information to the United States and the processing of that information as

described in this Agreement and the Privacy Policy.

6. TERM AND TERMINATION

This Agreement may be terminated (a) by your giving Altova written notice of termination; or (b) by Altova, at any time without prior notice. Upon any termination of the Agreement, you must cease all use of the Software that this Agreement governs, destroy all copies then in your possession or control and take such other actions as Altova may reasonably request to ensure that no copies of the Software remain in your possession or control. The terms and conditions set forth in Sections 1(c), 1(d), 1(e), 1(l), 2, 3, 5, 7, 8, and 9 survive termination as applicable.

7. RESTRICTED RIGHTS NOTICE AND EXPORT RESTRICTIONS

The Software was developed entirely at private expense and is commercial computer software provided with **RESTRICTED RIGHTS**. Use, duplication or disclosure by the U.S. Government or a U.S. Government contractor or subcontractor is subject to the restrictions set forth in this Agreement and as provided in FAR 12.211 and 12.212 (48 C.F.R. §12.211 and 12.212) or DFARS 227. 7202 (48 C.F.R. §227-7202) as applicable. Consistent with the above as applicable, Commercial Computer Software and Commercial Computer Documentation licensed to U.S. government end users only as commercial items and only with those rights as are granted to all other end users under the terms and conditions set forth in this Agreement. Manufacturer is Altova GmbH, Rudolfsplatz, 13a/9, A-1010 Vienna, Austria/EU. You may not use or otherwise export or re-export the Software or Documentation except as authorized by United States law and the laws of the jurisdiction in which the Software was obtained. In particular, but without limitation, the Software or Documentation may not be exported or re-exported (i) into (or to a national or resident of) any U.S. embargoed country or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce's Table of Denial Orders. By using the Software, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

8. U.S. GOVERNMENT ENTITIES

Notwithstanding the foregoing, if you are an agency, instrumentality or department of the federal government of the United States, then this Agreement shall be governed in accordance with the laws of the United States of America, and in the absence of applicable federal law, the laws of the Commonwealth of Massachusetts will apply. Further, and notwithstanding anything to the contrary in this Agreement (including but not limited to Section 5 (Indemnification)), all claims, demands, complaints and disputes will be subject to the Contract Disputes Act (41 U.S.C. §§7101 *et seq.*), the Tucker Act (28 U.S.C. §1346(a) and §1491), or the Federal Tort Claims Act (28 U.S.C. §§1346(b), 2401-2402, 2671-2672, 2674-2680), FAR 1.601(a) and 43.102 (Contract Modifications); FAR 12.302(b), as applicable, or other applicable governing authority. For the avoidance of doubt, if you are an agency, instrumentality, or department of the federal, state or local government of the U.S. or a U.S. public and accredited educational institution, then your indemnification obligations are only applicable to the extent they would not cause you to violate any applicable law (e.g., the Anti-Deficiency Act), and you have any legally required authorization or authorizing statute.

9. THIRD PARTY SOFTWARE

The Software may contain third party software which requires notices and/or additional terms and conditions. Such required third party software notices and/or additional terms and conditions are located at our Website at http://www.altova.com/legal_3rdparty.html and are made a part of and incorporated by reference into this Agreement. By accepting this Agreement, you are also

accepting the additional terms and conditions, if any, set forth therein.

10. GENERAL PROVISIONS

If you are located in the European Union and are using the Software in the European Union and not in the United States, then this Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht, Wien (Commercial Court, Vienna) in connection with any such dispute or claim.

If you are located in the United States or are using the Software in the United States then this Agreement will be governed by and construed in accordance with the laws of the Commonwealth of Massachusetts, USA (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the federal or state courts of the Commonwealth of Massachusetts and you further agree and expressly consent to the exercise of personal jurisdiction in the federal or state courts of the Commonwealth of Massachusetts in connection with any such dispute or claim.

If you are located outside of the European Union or the United States and are not using the Software in the United States, then this Agreement will be governed by and construed in accordance with the laws of the Republic of Austria (excluding its conflict of laws principles and the U.N. Convention on Contracts for the International Sale of Goods) and you expressly agree that exclusive jurisdiction for any claim or dispute with Altova or relating in any way to your use of the Software resides in the Handelsgericht, Wien (Commercial Court, Vienna) and you further agree and expressly consent to the exercise of personal jurisdiction in the Handelsgericht Wien (Commercial Court, Vienna) in connection with any such dispute or claim. This Agreement will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

This Agreement contains the entire agreement and understanding of the parties with respect to the subject matter hereof, and supersedes all prior written and oral understandings of the parties with respect to the subject matter hereof. Any notice or other communication given under this Agreement shall be in writing and shall have been properly given by either of us to the other if sent by certified or registered mail, return receipt requested, or by overnight courier to the address shown on Altova's Web site for Altova and the address shown in Altova's records for you, or such other address as the parties may designate by notice given in the manner set forth above. This Agreement will bind and inure to the benefit of the parties and our respective heirs, personal and legal representatives, affiliates, successors and permitted assigns. The failure of either of us at any time to require performance of any provision hereof shall in no manner affect such party's right at a later time to enforce the same or any other term of this Agreement. This Agreement may be amended only by a document in writing signed by both of us. In the event of a breach or threatened breach of this Agreement by either party, the other shall have all applicable equitable as well as legal remedies. Each party is duly authorized and empowered to enter into and perform this Agreement. If, for any reason, any provision of this Agreement is held invalid or otherwise unenforceable, such invalidity or unenforceability shall not affect the remainder of this Agreement, and this Agreement shall continue in full force and effect to the fullest extent allowed by law. The parties knowingly and expressly consent to the foregoing terms and conditions.

Last updated: 2015-11-01

Index

$

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X