All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

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

Published: 2021

© 2015-2021 Altova GmbH
# Table of Contents

1 Welcome to MobileTogether Designer ................................................................................................................. 21

2 New Features .......................................................................................................................................................... 23
   2.1 Version 6 .................................................................................................................................................. 28
   2.2 Version 5 .................................................................................................................................................. 30
   2.3 Version 4 .................................................................................................................................................. 35
   2.4 Version 3 .................................................................................................................................................. 39
   2.5 Version 2 .................................................................................................................................................. 42
   2.6 Version 1 .................................................................................................................................................. 45

3 Introduction .......................................................................................................................................................... 47
   3.1 MobileTogether Overview ...................................................................................................................... 48
   3.2 Terminology Q&A .................................................................................................................................. 50
   3.3 Design Steps .......................................................................................................................................... 51
   3.4 XPath in MobileTogether ...................................................................................................................... 53

4 Tutorials ............................................................................................................................................................... 56
   4.1 QuickStart (Part 1) .................................................................................................................................. 58
       4.1.1 Create a New Design ...................................................................................................................... 58
       4.1.2 Set Up a Page ................................................................................................................................ 60
       4.1.3 Add a Page Source (or Data Source) .......................................................................................... 61
       4.1.4 Format the Design .......................................................................................................................... 64
       4.1.5 Add a Control: Combo Box ........................................................................................................... 65
       4.1.6 Add a Control: Image ..................................................................................................................... 67
       4.1.7 Define Control Actions ................................................................................................................. 70
       4.1.8 Validate the Project ....................................................................................................................... 73
       4.1.9 Run a Simulation ............................................................................................................................ 73
## QuickStart (Part 2)

1. **Load Data from a File**
2. **Change Source Node**
3. **Run a Simulation**
4. **Use File Data for Combo Box Entries**
5. **Set Data File as Default File**
6. **Create Dynamic Links to Web Pages**
7. **Save Data Back to File**

## Simple Database

1. **The DB Data Source**
2. **Persistent Tree to Hold User Input**
3. **Load DB Data Based on User Selection**

## Advanced Database

1. **Page Sources: DB and Persistent**
2. **Controlling Visibility**
3. **Images in the DB**
4. **Editing Records**
5. **Adding Records**

## Database-And-Charts

1. **The Project Structure**
2. **The Main Page**
3. **Page Sources of the Main Page**
4. **The Combo Boxes**
5. **The Tabular Report**
6. **The Charts**
7. **Edit Offices Table**
8. **Edit Sales Table**

## SubPages-And-Visibility

1. **Design Structure**
2. **Page Source Listings**
3. **Top Page: Page Sources**
4. **Top Page: Customers Table**
5. **Top Page: Action Group, Go to Sub Page**
6. **Top Page: Show All Orders Action**

### Table of Contents

- QuickStart (Part 2) ................................................................. 79
- Simple Database ............................................................... 93
- Advanced Database ......................................................... 103
- Database-And-Charts ..................................................... 119
- SubPages-And-Visibility .................................................. 148

### Index

- Deploy to Server .......................................................... 76
- Load Data from a File .................................................. 79
- Change Source Node .................................................... 82
- Run a Simulation ......................................................... 84
- Use File Data for Combo Box Entries ......................... 85
- Set Data File as Default File ........................................ 87
- Create Dynamic Links to Web Pages ......................... 89
- Save Data Back to File ............................................... 91
- The DB Data Source .................................................... 93
- Persistent Tree to Hold User Input ............................... 95
- Load DB Data Based on User Selection ..................... 97
- Page Sources: DB and Persistent ............................... 105
- Controlling Visibility .................................................. 110
- Images in the DB ....................................................... 111
- Editing Records .......................................................... 114
- Adding Records ......................................................... 116
- The Project Structure .................................................. 120
- The Main Page ............................................................ 121
- Page Sources of the Main Page ................................. 123
- The Combo Boxes ....................................................... 127
- The Tabular Report ...................................................... 129
- The Charts ................................................................. 130
- Edit Offices Table ........................................................ 135
- Edit Sales Table .......................................................... 141
- Design Structure .......................................................... 149
- Page Source Listings ................................................... 150
- Top Page: Page Sources ............................................. 152
- Top Page: Customers Table ......................................... 154
- Top Page: Action Group, Go to Sub Page .................. 156
- Top Page: Show All Orders Action ............................. 158
4.6.7 Sub Page: Page Sources .......................................................... 159
4.6.8 Sub Page: Orders Table .......................................................... 160
4.6.9 Sub Page: Visibility Property .................................................. 162
4.6.10 Sub Page: Decimal Totals in XPath ...................................... 163
4.6.11 Simulation and Testing .......................................................... 164

4.7 Add and Edit Records ................................................................. 166
  4.7.1 Design Pages ................................................................. 166
  4.7.2 Page Sources ................................................................. 168
  4.7.3 Add a New Record .......................................................... 169
  4.7.4 Enter New-Record Data .................................................... 171
  4.7.5 Display All Records ......................................................... 171
  4.7.6 Edit an Existing Record .................................................... 174

4.8 SOAP Requests .................................................................... 175
  4.8.1 The XML Page Source ...................................................... 177
  4.8.2 Design Components ......................................................... 181
  4.8.3 Refreshing the Page ........................................................ 186

4.9 Sharing Geolocations ............................................................. 190
  4.9.1 Reading and Sharing the Geolocation ............................... 191
  4.9.2 Using Try/Catch/Throw Exceptions ................................... 193

4.10 Scrollable Tables ................................................................. 196
  4.10.1 Tables that Force Full Screen Height ............................... 197
  4.10.2 Tables Having Specific Heights ....................................... 198

5 User Interface ........................................................................ 201

  5.1 Main Window ...................................................................... 203
    5.1.1 Page Design .............................................................. 204
    5.1.2 DB Query ................................................................. 205

  5.2 Pages Pane .................................................................... 207

  5.3 Files Pane .................................................................... 209

  5.4 Controls Pane .................................................................. 211

  5.5 Breakpoints Pane ............................................................. 214

  5.6 Page Sources Pane .......................................................... 215

  5.7 Overview Pane ................................................................. 217

  5.8 Styles & Properties Pane .................................................... 219
5.9 Messages Pane................................................................................................................... 223
5.10 Listings Pane................................................................................................................... 225
5.11 Find & Replace Pane....................................................................................................... 227

6 Project .................................................................................................................................. 230
6.1 Client-Server Interaction................................................................................................. 231
6.2 Location of Project Files................................................................................................. 233
6.3 Deploying the Project....................................................................................................... 235
6.4 MobileTogether Packages............................................................................................... 239
6.5 Project Properties............................................................................................................ 240
6.6 Localization....................................................................................................................... 250
6.7 Namespaces..................................................................................................................... 252
6.8 Global Resources............................................................................................................. 253
6.9 Performance..................................................................................................................... 254
  6.9.1 Embed XML in Design File......................................................................................... 254
  6.9.2 Data Querying with XQuery 3.1................................................................................. 254
  6.9.3 Data Storage on Servers......................................................................................... 254
  6.9.4 Persistent Data Storage on Clients.......................................................................... 256

7 Page Sources (Data Sources) .............................................................................................. 257
7.1 Types of Page Sources: Adding....................................................................................... 259
  7.1.1 XML Sources............................................................................................................ 261
  7.1.2 HTML Sources........................................................................................................ 263
  7.1.3 JSON Sources........................................................................................................... 265
  7.1.4 HTTP Sources.......................................................................................................... 267
  7.1.5 DB Sources.............................................................................................................. 281
  7.1.6 XQuery Sources.................................................................................................... 282
  7.1.7 FlowForce Jobs..................................................................................................... 284
7.2 Page Source Properties..................................................................................................... 287
7.3 Page Source Trees............................................................................................................ 289
  7.3.1 Root Nodes............................................................................................................... 291
  7.3.2 Namespaces in the Project.................................................................................... 293
  7.3.3 Tree Structure......................................................................................................... 294
7.4 Caches ...................................................................................................................................... 316
  7.4.1 Creating Caches ................................................................................................................... 317
  7.4.2 Cache Overview .................................................................................................................... 318

8 Pages and Page Events ................................................................................................................. 320
  8.1 Pages, Tabbed Pages, and Sub Pages ......................................................................................... 321
  8.2 Data Sources of a Page .............................................................................................................. 323
  8.3 Page Properties ........................................................................................................................ 324
  8.4 Page Events .............................................................................................................................. 329
    8.4.1 OnPageLoad ......................................................................................................................... 330
    8.4.2 OnPageRefresh .................................................................................................................... 331
    8.4.3 OnBackButtonClicked ......................................................................................................... 333
    8.4.4 OnSubmitButtonClicked ..................................................................................................... 334
    8.4.5 OnServerConnectionError ................................................................................................. 334
    8.4.6 OnEmbeddedMessage ....................................................................................................... 337

9 Controls and Control Events ........................................................................................................ 340
  9.1 Controls .................................................................................................................................. 341
    9.1.1 Assertion Message .............................................................................................................. 345
    9.1.2 Button ................................................................................................................................. 352
    9.1.3 Chart .................................................................................................................................. 371
    9.1.4 Check Box ............................................................................................................................ 379
    9.1.5 Combo Box .......................................................................................................................... 390
    9.1.6 Date .................................................................................................................................... 403
    9.1.7 DateTime (iOS) ...................................................................................................................... 414
    9.1.8 Edit Field ............................................................................................................................. 424
    9.1.9 Geolocation Map ............................................................................................................... 436
    9.1.10 Horizontal Line .................................................................................................................. 443
    9.1.11 Horizontal Slider ............................................................................................................... 448
    9.1.12 Image .................................................................................................................................. 455
    9.1.13 Label .................................................................................................................................. 467
10.2.6 Audio Recording .................................................................................................................. 613
9.2 Control Events .......................................................................................................................... 572

10 Actions ....................................................................................................................................... 574

10.1 User Interactions ....................................................................................................................... 578
  10.1.1 Message Box .......................................................................................................................... 580
  10.1.2 Send Email To ....................................................................................................................... 582
  10.1.3 Share ................................................................................................................................... 587
  10.1.4 Send SMS To ......................................................................................................................... 590
  10.1.5 Make Call To ........................................................................................................................ 591
  10.1.6 Open URL/File ..................................................................................................................... 591
  10.1.7 Print To ................................................................................................................................ 596
  10.1.8 MapForce Transfer .......................................................................................................... 602
  10.1.9 Wait Cursor ........................................................................................................................ 605
  10.1.10 Read Contacts .................................................................................................................. 605
  10.1.11 Access Calendar ............................................................................................................... 607
  10.1.12 Let User Choose Date ....................................................................................................... 611
  10.1.13 Let User Choose Time ..................................................................................................... 612

10.2 Images, Audio, Video ............................................................................................................... 613
  10.2.1 Let User Choose Image ...................................................................................................... 615
  10.2.2 Load/Save Image ................................................................................................................ 616
  10.2.3 View Image ........................................................................................................................ 622
  10.2.4 Let User Scan Barcode ...................................................................................................... 623
  10.2.5 Audio ................................................................................................................................ 624
  10.2.6 Audio Recording .................................................................................................................. 628
10.8.6 Load/Save String.................................................................................................................. 710
10.8.7 Load from SOAP.................................................................................................................... 713
10.8.8 Save ...................................................................................................................................... 715
10.8.9 Delete File/Folder................................................................................................................... 717
10.8.10 Reset .................................................................................................................................... 718
10.8.11 Execute SOAP Request...................................................................................................... 719
10.8.12 Execute REST Request....................................................................................................... 721
10.8.13 Execute FlowForce Job...................................................................................................... 722
10.8.14 Get File Info....................................................................................................................... 723
10.8.15 Read Folder......................................................................................................................... 725
10.8.16 Backup/Restore Page Sources............................................................................................ 727

10.9 Database.................................................................................................................................. 728
10.9.1 DB Begin Transaction.......................................................................................................... 730
10.9.2 DB Execute.......................................................................................................................... 732
10.9.3 DB Bulk Insert Into.............................................................................................................. 735
10.9.4 DB Commit Transaction....................................................................................................... 737
10.9.5 DB Rollback Transaction...................................................................................................... 739
10.9.6 DB Read Structure................................................................................................................ 740
10.9.7 Backup/Restore SQLite DB.................................................................................................. 743

10.10 Miscellaneous....................................................................................................................... 746
10.10.1 Comment............................................................................................................................. 748
10.10.2 Execute On.......................................................................................................................... 748
10.10.3 Execute at Once.................................................................................................................. 750
10.10.4 Cancel Action Execution.................................................................................................... 750
10.10.5 User Cancel Behavior........................................................................................................ 751
10.10.6 Solution Execution............................................................................................................. 753
10.10.7 Set Language...................................................................................................................... 754
10.10.8 Measure Controls............................................................................................................... 756
10.10.9 Embedded Message Back................................................................................................. 758
10.10.10 Log Message...................................................................................................................... 758

10.11 Update Data.......................................................................................................................... 760
10.11.1 Update Node(s).................................................................................................................... 762
10.11.2 Insert Node(s)...................................................................................................................... 767
10.11.3 Append Node(s).................................................................................................................. 770
10.11.4 Delete Node(s)..................................................................................................................... 775

Altova MobileTogether Designer
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.11.5</td>
<td>Replace Node(s)</td>
<td>776</td>
</tr>
<tr>
<td>10.12</td>
<td>If, Loop, Let, Try/Catch, Throw</td>
<td>780</td>
</tr>
<tr>
<td>10.12.1</td>
<td>If-Then</td>
<td>782</td>
</tr>
<tr>
<td>10.12.2</td>
<td>If-Then-Else</td>
<td>782</td>
</tr>
<tr>
<td>10.12.3</td>
<td>Loop</td>
<td>783</td>
</tr>
<tr>
<td>10.12.4</td>
<td>Break Loop</td>
<td>785</td>
</tr>
<tr>
<td>10.12.5</td>
<td>Let</td>
<td>786</td>
</tr>
<tr>
<td>10.12.6</td>
<td>Update Variable</td>
<td>789</td>
</tr>
<tr>
<td>10.12.7</td>
<td>Throw</td>
<td>789</td>
</tr>
<tr>
<td>10.12.8</td>
<td>Try/Catch Exceptions</td>
<td>791</td>
</tr>
<tr>
<td>10.12.9</td>
<td>Try/Catch Server Connection Errors</td>
<td>793</td>
</tr>
<tr>
<td>10.12.10</td>
<td>Return</td>
<td>793</td>
</tr>
<tr>
<td>10.13</td>
<td>Action Groups</td>
<td>795</td>
</tr>
<tr>
<td>10.13.1</td>
<td>Managing Action Groups</td>
<td>797</td>
</tr>
<tr>
<td>10.13.2</td>
<td>Action Groups for Reusing Actions</td>
<td>798</td>
</tr>
<tr>
<td>10.13.3</td>
<td>Action Groups with Parameters</td>
<td>799</td>
</tr>
<tr>
<td>10.13.4</td>
<td>Action Groups with Action-Group Parameters</td>
<td>801</td>
</tr>
<tr>
<td>10.13.5</td>
<td>Variables and Action Group Results</td>
<td>806</td>
</tr>
</tbody>
</table>

## 11 Design Components 808

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1</td>
<td>Tables</td>
<td>809</td>
</tr>
<tr>
<td>11.1.1</td>
<td>Static Tables</td>
<td>810</td>
</tr>
<tr>
<td>11.1.2</td>
<td>Repeating Tables</td>
<td>812</td>
</tr>
<tr>
<td>11.1.3</td>
<td>Dynamic Rows</td>
<td>816</td>
</tr>
<tr>
<td>11.1.4</td>
<td>Dynamic Columns</td>
<td>820</td>
</tr>
<tr>
<td>11.1.5</td>
<td>Table Properties</td>
<td>824</td>
</tr>
<tr>
<td>11.1.6</td>
<td>Table Context Menu</td>
<td>831</td>
</tr>
<tr>
<td>11.2</td>
<td>Images</td>
<td>834</td>
</tr>
<tr>
<td>11.2.1</td>
<td>Image Source</td>
<td>834</td>
</tr>
<tr>
<td>11.2.2</td>
<td>Base64-Encoded Images</td>
<td>835</td>
</tr>
<tr>
<td>11.2.3</td>
<td>Exchangeable Image File Format (Exif)</td>
<td>838</td>
</tr>
<tr>
<td>11.2.4</td>
<td>Images Chosen by End User</td>
<td>844</td>
</tr>
<tr>
<td>11.2.5</td>
<td>Transforming Images</td>
<td>850</td>
</tr>
<tr>
<td>11.2.6</td>
<td>Images in Databases</td>
<td>851</td>
</tr>
</tbody>
</table>
11.3 Audio, Video ................................................................. 852
  11.3.1 Audio Playback ...................................................... 852
  11.3.2 Audio Recording .................................................... 854
  11.3.3 Text to Speech ....................................................... 856
  11.3.4 Video Playback ...................................................... 856
  11.3.5 Video Recording ..................................................... 859
  11.3.6 Audio/Video Formats ............................................. 860
11.4 NFC ........................................................................... 862
  11.4.1 Discovering and Reading NFC Tags ....................... 864
  11.4.2 Pushing Data to Other Devices ............................... 865
  11.4.3 NFC-Related Events .............................................. 866
  11.4.4 Design Components for NFC ................................. 867
11.5 Push Notifications ..................................................... 869
  11.5.1 The Sending Solution ............................................ 870
  11.5.2 The Receiving Solution ......................................... 872
  11.5.3 Push Notifications in AppStore Apps ....................... 874
  11.5.4 Simulating Push Notifications ................................. 877
11.6 Charts ....................................................................... 879
  11.6.1 Creating and Configuring Charts .............................. 879
  11.6.2 Chart Data Selection ............................................. 882
  11.6.3 Chart Settings and Appearance .............................. 901
11.7 Control Templates ..................................................... 930
  11.7.1 Creating a Control Template ................................... 931
  11.7.2 Using a Control Template: Placeholder Controls .......... 935
  11.7.3 Overriding Template Events ................................. 937
  11.7.4 Example Projects ............................................... 939
11.8 Style Sheets ............................................................ 954
  11.8.1 Style Sheet Type and Scope .................................... 956
  11.8.2 Priority within a Style Sheet ................................. 957
  11.8.3 Priority across Style Sheets ................................. 961
  11.8.4 Applying User-Created Style Sheets ....................... 963
  11.8.5 Style Sheet Properties ......................................... 964
11.9 Rich Text ................................................................. 966
  11.9.1 The Rich Text Control .......................................... 966
  11.9.2 Rich Text Style Sheets: Setup ............................... 967
11.9.3 Rich Text Style Sheets: Styles .......................................................... 969
11.9.4 Editing Rich Text Content ................................................................. 974
11.10 Solutions for Authenticated Users ...................................................... 978
11.11 Hyperlinking to Solutions .................................................................. 980
11.12 Sizes: Pixels, DPI, DP, SP ................................................................. 984

12 XPath/XQuery Expressions and Functions ............................................. 987
12.1 XPath/XQuery Window ....................................................................... 988
  12.1.1 Expression Builder ......................................................................... 989
  12.1.2 Expression Evaluator ................................................................... 992
  12.1.3 XPath Debugger ........................................................................... 996
12.2 MobileTogether Extension Functions ................................................... 1006
12.3 User-Defined XPath/XQuery Functions ................................................. 1035
12.4 FAQ about XPath/XQuery ................................................................. 1038

13 Global Variables ..................................................................................... 1040
13.1 Static Global Variables ....................................................................... 1042
13.2 Dynamic Local Variables ................................................................... 1046
13.3 User Variables ................................................................................... 1050

14 Databases .............................................................................................. 1052
14.1 DBs as Data Sources .......................................................................... 1054
14.2 Connecting to a Data Source .............................................................. 1058
   14.2.1 Starting the Connection Wizard ...................................................... 1059
   14.2.2 Database Drivers Overview ........................................................... 1061
   14.2.3 Setting up an ADO Connection ....................................................... 1064
   14.2.4 Setting up an ADO.NET Connection ............................................... 1070
   14.2.5 Setting up an ODBC Connection .................................................... 1077
   14.2.6 Setting up a JDBC Connection ....................................................... 1080
   14.2.7 Setting up a PostgreSQL Connection ............................................. 1084
   14.2.8 Setting up a SQLite Connection ...................................................... 1086
   14.2.9 Using a Connection from Global Resources ..................................... 1087
   14.2.10 Database Connection Examples .................................................... 1088
14.3 Selecting DB Objects as Data Sources................................................................. 1143
14.4 Editing DB Data.................................................................................................. 1146
14.5 Saving Data to the DB....................................................................................... 1150
14.6 The DB Execute Action..................................................................................... 1154
14.7 Displaying DB Data............................................................................................ 1158
14.8 Database Query.................................................................................................. 1160
   14.8.1 GUI Overview and Toolbar........................................................................ 1162
   14.8.2 Connecting to Data Sources........................................................................ 1164
   14.8.3 Browser Pane............................................................................................... 1166
   14.8.4 Query Pane: Description............................................................................ 1169
   14.8.5 Query Pane: Working With.......................................................................... 1173
   14.8.6 Results and Messages Pane.......................................................................... 1173

15 Altova Global Resources ..................................................................................... 1176
15.1 Defining Global Resources................................................................................ 1177
   15.1.1 Files........................................................................................................... 1179
   15.1.2 Folders....................................................................................................... 1182
   15.1.3 Databases.................................................................................................. 1183
15.2 Using Global Resources.................................................................................... 1186
   15.2.1 Assigning Files and Folders....................................................................... 1186
   15.2.2 Assigning Databases................................................................................ 1187
   15.2.3 Changing the Active Configuration........................................................... 1188

16 Simulation ........................................................................................................... 1190
16.1 Simulation in MobileTogether Designer............................................................ 1191
16.2 Simulation on Server........................................................................................ 1196
16.3 Simulation on Client.......................................................................................... 1203
16.4 Geolocation Settings......................................................................................... 1205
16.5 NFC Sample Files............................................................................................. 1210
16.6 Push Notification Simulations.......................................................................... 1212
16.7 Contacts Sample Files...................................................................................... 1214
16.8 Calendar Sample Files...................................................................................... 1215
16.9 Service Trigger Simulations............................................................................... 1216
16.10 Messages Pane.......................................................................................................................................................... 1218

17 Offline Usage 1221
17.1 Enter Data Offline and Upload................................................................. 1223
17.2 Connect to Server on Demand................................................................. 1225
17.3 Edit Data Offline and Synchronize........................................................ 1228

18 MT Debugger 1232
18.1 Debugger Modes...................................................................................... 1233
18.2 Actions Debugger.................................................................................... 1234
18.3 XPath Debugger....................................................................................... 1242

19 Embedded Webpage Solutions 1243
19.1 Embedding a Solution in a Webpage....................................................... 1247
19.2 Communication between Webpage and Server................................. 1250
  19.2.1 Posting: From Webpage to Server................................................... 1251
  19.2.2 Listening: From Server to Webpage............................................... 1252
19.3 Authentication......................................................................................... 1254
  19.3.1 Anonymous Login.......................................................................... 1254
  19.3.2 User Login..................................................................................... 1255
  19.3.3 JWT Authentication....................................................................... 1255
19.4 Examples.............................................................................................. 1261
  19.4.1 Embedding a Solution..................................................................... 1261
  19.4.2 Sending/Receiving JSON Data....................................................... 1261
  19.4.3 Sending/Receiving XML Data....................................................... 1269
  19.4.4 Pre-setting the JSON Page Source................................................ 1277
  19.4.5 JWT Authentication...................................................................... 1283

20 Automated Testing 1287
20.1 Recording a Test Case............................................................................ 1289
20.2 Playing Back a Test Case........................................................................ 1291
20.3 Managing Test Cases and Runs............................................................ 1294
20.4 Deploying Test Cases to Server ................................................................. 1298
20.5 Comparing Test Runs .............................................................................. 1302

21 AppStore Apps ......................................................................................... 1304
21.1 Generate Program Code from Project .................................................. 1305
21.2 Deploy Workflow to Server .................................................................... 1316
21.3 Compile Program Code ........................................................................ 1318
  21.3.1 Android.......................................................................................... 1318
  21.3.2 iOS ............................................................................................... 1319
  21.3.3 Windows App ............................................................................... 1321
21.4 SPL Templates ...................................................................................... 1324
  21.4.1 SPL Syntax ................................................................................... 1325
  21.4.2 String Mechanisms ....................................................................... 1327
  21.4.3 Properties of $Application ............................................................ 1331
  21.4.4 Miscellaneous Objects ................................................................. 1332

22 In-App Purchases .................................................................................... 1333
22.1 Register Products ................................................................................ 1335
22.2 In-App-Purchase Page Source ............................................................. 1337
22.3 Query Available Products ..................................................................... 1339
22.4 Purchase Products .............................................................................. 1341
22.5 Other Operations ................................................................................ 1343
22.6 Simulation and Testing ........................................................................ 1346
22.7 Example Project .................................................................................. 1347
  22.7.1 Map Product IDs to Product Name ................................................ 1349
  22.7.2 The $PERSISTENT Page Source .................................................. 1350
  22.7.3 Availability of In-App Service ....................................................... 1350
  22.7.4 Get Available Products ............................................................... 1351
  22.7.5 The "Buy" Button ......................................................................... 1354
  22.7.6 The OnPurchaseUpdated Event .................................................... 1358
  22.7.7 Query/Restore Purchases .............................................................. 1359

23 MT Solutions in UWP Apps .................................................................. 1363

Altova MobileTogether Designer
25.4 Debug

25.4.1 Resume Debugging / Go

25.4.2 Stop Debugging

25.4.3 Step into Action
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.4.4</td>
<td>Step into XPath</td>
<td>1435</td>
</tr>
<tr>
<td>25.4.5</td>
<td>Step Out</td>
<td>1436</td>
</tr>
<tr>
<td>25.4.6</td>
<td>Step Over</td>
<td>1436</td>
</tr>
<tr>
<td>25.4.7</td>
<td>Run to Selected Action</td>
<td>1436</td>
</tr>
<tr>
<td>25.4.8</td>
<td>Stop at Next Error</td>
<td>1436</td>
</tr>
<tr>
<td>25.4.9</td>
<td>Stop at Next Breakpoint</td>
<td>1437</td>
</tr>
<tr>
<td>25.4.10</td>
<td>Stop at Next Action</td>
<td>1437</td>
</tr>
<tr>
<td>25.4.11</td>
<td>Disable All Breakpoints</td>
<td>1438</td>
</tr>
<tr>
<td>25.4.12</td>
<td>Enable All Breakpoints</td>
<td>1438</td>
</tr>
<tr>
<td>25.4.13</td>
<td>Remove All Breakpoints</td>
<td>1438</td>
</tr>
<tr>
<td>25.4.14</td>
<td>Debug Windows</td>
<td>1438</td>
</tr>
<tr>
<td>25.5</td>
<td>Page</td>
<td>1439</td>
</tr>
<tr>
<td>25.5.1</td>
<td>Page Actions</td>
<td>1440</td>
</tr>
<tr>
<td>25.5.2</td>
<td>Actions Overview</td>
<td>1441</td>
</tr>
<tr>
<td>25.5.3</td>
<td>Jump to Control</td>
<td>1442</td>
</tr>
<tr>
<td>25.5.4</td>
<td>Show/Define Tab Order</td>
<td>1442</td>
</tr>
<tr>
<td>25.5.5</td>
<td>List Text Size Auto-Fit Groups</td>
<td>1444</td>
</tr>
<tr>
<td>25.6</td>
<td>Table</td>
<td>1445</td>
</tr>
<tr>
<td>25.6.1</td>
<td>Insert/Delete Table</td>
<td>1445</td>
</tr>
<tr>
<td>25.6.2</td>
<td>Insert/Append/Delete Row/Column</td>
<td>1446</td>
</tr>
<tr>
<td>25.6.3</td>
<td>Join/Split Cells</td>
<td>1447</td>
</tr>
<tr>
<td>25.6.4</td>
<td>Show Add–Remove Buttons</td>
<td>1447</td>
</tr>
<tr>
<td>25.6.5</td>
<td>Add Header/Footer, Leading/Trailing Column</td>
<td>1448</td>
</tr>
<tr>
<td>25.6.6</td>
<td>Remove Header/Footer, Leading/Trailing Column</td>
<td>1449</td>
</tr>
<tr>
<td>25.6.7</td>
<td>Convert Row to Repeating/Static Row</td>
<td>1451</td>
</tr>
<tr>
<td>25.6.8</td>
<td>Convert to Repeating/Non-Repeating Table</td>
<td>1451</td>
</tr>
<tr>
<td>25.6.9</td>
<td>Convert Column to Repeating/Static Column</td>
<td>1451</td>
</tr>
<tr>
<td>25.6.10</td>
<td>Border Settings</td>
<td>1452</td>
</tr>
<tr>
<td>25.7</td>
<td>View</td>
<td>1458</td>
</tr>
<tr>
<td>25.7.1</td>
<td>Status Bar and Panes</td>
<td>1458</td>
</tr>
<tr>
<td>25.7.2</td>
<td>Zoom Levels</td>
<td>1458</td>
</tr>
<tr>
<td>25.8</td>
<td>Tools</td>
<td>1460</td>
</tr>
<tr>
<td>25.8.1</td>
<td>Global Resources</td>
<td>1460</td>
</tr>
<tr>
<td>25.8.2</td>
<td>Active Configuration</td>
<td>1461</td>
</tr>
<tr>
<td>25.8.3</td>
<td>User-Defined Tools</td>
<td>1462</td>
</tr>
</tbody>
</table>
1 Welcome to MobileTogether Designer

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

Current version: 7.3

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:

- New Features
- Introduction
- Tutorials
- User Interface
- Project
- Page Sources (Data Sources)
- Pages and Page Events
- Controls and Control Events
- Actions
- Design Components
- XPath/XQuery Expressions and Functions
- Global Variables
- Databases
- Altova Global Resources
- Simulation
- Offline Usage
- Embedded Webpage Solutions
- Automated Testing
- AppStore Apps
- In-App Purchases
- MT Solutions in UWP Apps
- Server Services
- Menu Commands
- Frequently Asked Questions
- Appendices

Last updated: 30 June 2021

Altova website: App development, Enterprise apps, Enterprise app development, RMAD, Low code app development
2 New Features

Version 7.3

In-app Purchases

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

AppStore apps and their simulations

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

MobileTogether solutions in UWP apps

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

Actions

- New actions to implement the In-app Purchases mechanism: Purchase, Restore Purchases, Query Purchases, Query Available Products, Acknowledge Purchase, Get/Report Consumable.
New Features

- The View Image action has been extended with an auto-rotate property that enables the viewed image to be auto-rotated.
- The Open URL action can be used to construct a command line instruction that can be used in designer simulations.

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

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

Version 7.2

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

Page sources
- In the Page Sources window, a new context menu command of root nodes deletes the selected page source from all pages of the project.
- In the Project menu, the command List Page Sources by Attribute lists the project's page sources into groups according to the values of their attributes.
New Features

- The default XML file of XML page sources can be edited directly in Altova XMLSpy by selecting the View Default File in XMLSpy command.

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

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

Version 7.1

- Internal updates

Version 7.0

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

**Page properties**
- Length units of design components and text size in previous versions were given either as pixels or a percentage of a containing component. From version 7.0 onwards, page-related length units can be also be specified as device-independent pixels (dp) and scale-independent pixels (sp). The built-in XPath extension function mt-convert-units enables you to convert between units.
- The Browser Width property has been renamed to Browser Max Width, and now specifies the maximum width of a solution's page in the browser.
**New Features**

- A **page's properties** now enables page margins to be set.
- A new property named **All Styles** enables you to set all **page styling properties** in one convenient location via an XPath map expression.

**Controls**

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

**Actions**

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

**XPath-related features and XPath extension functions**

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

- The browser settings of projects have been extended with two new settings: (i) a base size for calculating the font size of controls; (ii) whether back navigation in solutions that are embedded in IFrames is allowed or not.
- You can specify the default starting language of a multi-language solution when you deploy the solution.
- User Variables can be defined to be stored on the client only, which improves speed when the variable contains or involves the use of large datasets.
- Support for Android 10.
2.1 Version 6

Version 6.0

Control templates and Placeholder Controls

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

Controls

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

Actions

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

Action Groups

- You can now also pass arrays and maps to Action Groups as parameter values. This is in addition to being able to pass atomic values and nodesets.
- In the Execute Action Group action, the Action Group to execute can be selected via an XPath expression, thus enabling the Action Group to be selected dynamically. This option is in addition to
being able to select the Action Group from a dropdown list that contains all the Action Groups defined in the design (a fixed selection).

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

**JSON**

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

**XPath-related features and XPath extension functions**

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

**Databases**

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

**Miscellaneous**

- If you want the user of a solution to go to a solution on another MobileTogether Server, then you can pass authentication information securely to the second server, thus obviating the need for a second user login.
- A new Java tab in the Options dialog to specify the location of the Java VM (virtual machine) on your system.
- **Parameters and variables can be declared on a sub page**. This provides you with more flexibility for dealing dynamically with data on the sub page.
- In the project settings, you can select the theme of the solution: light, dark, or device-selected.
- On Android 7 devices and newer, the app window has been configured for use in multi-window display (aka split-screen mode).
2.2 Version 5

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

Version 5.4

New features and updates in MobileTogether Designer Version 5.4:

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

Version 5.3

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

Version 5.2

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

Version 5.1

New features and updates in MobileTogether Designer Version 5.1:

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

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

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

**Version 5.0**

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

**Controls**
- New border properties (width, color, and style) have been introduced for table items (cells, columns, rows, and table).
- Border settings can be applied quickly to different table items by using the Border Settings dialog.
- Shorthand margin and padding properties have been added to controls that have margin and padding properties. This enables a common value for all four sides to be specified in a single property, rather than individually for each side.
Padding properties have been added to more controls where padding is applicable. This enables space to be added between individual borders of a control (left, right, top, and bottom) and the control’s content.

In a Combo Box control that allows multiple values to be selected by the end user, different separators can be used to construct (i) the string that is entered in the page source node and (ii) the string that is displayed in the combo box. The former is set via the Multi Select Separator property, the latter via the Multi Select Separator Visible property.

In Combo Boxes, the text of dropdown list items and the corresponding XML data values can be defined via an XPath expression that returns a sequence of two-member arrays. This way of defining combo box entries is in addition to existing options.

The appearance of Buttons has been enhanced so that both a Button image (see the Button Image property) as well as a Button text can be displayed. For the Button image, you can choose from a range of predefined icons or use a custom image. You can also set the position of the Button image to be to the left or the right of the Button text.

The Text Size Auto-Fit property of Controls that have a Text Size property has an additional option. Text that is too long to fit within the width of a control is shown as a truncated string that ends with an ellipsis. This is in addition to existing options for auto-fitting text.

Values for the Checked Values (true/false) property of the Check Box and Switch controls can also be set in style sheets—which enables this property’s values to be set globally for these controls.

The Checked Values property of the Radio Button control can also be set in style sheets. This enables this property’s checked value to be set globally.

The OnEnter/OnEscape events of the Button, Chart, Image, and Label controls have been enhanced for use on all client devices (in addition to their use on Web and Windows clients).

The Rich Text control can use a predefined Rich Text style sheet for text that is marked up with HTML tags.

The text that will be the content of the control can be defined with an XPath expression that evaluates to an HTML-encoded string.

The Vertical Line control can be given top and bottom margins.

The Horizontal Slider control has an Auto Correct Value property that automatically corrects values in the associated page source node to a value that is within the defined slider value range.

A context menu command for controls in the Controls Pane displays all instances of that control type.

Actions

A new Quick Filter text box in the Actions dialog enables you to filter the actions and action groups of the dialog.

Actions and action groups can be added to the actions of an event from a popup in the event pane. This is in addition to the standard way of adding an action by dragging it into the event pane.

A new action, Load/Save Text File, enables (i) text to be loaded from a file to a page source node, and (ii) for text to be saved from a page source node to a text file.

A new action, DB Read Structure, enables the structure of a DB to be read and for data in the DB to be stored in a new type of page source, the $MT_DBSTRUCTURE page source. Data in this page source can be used in the same way as data in any other page source. This action is useful if you want only a data read-out.

The Open URL/File action has been enhanced to accept data URLs. As a result, binary files can be opened directly in a new tab of the web client’s browser.

The Update Node(s) action enables multiple nodes to be updated by specifying the target nodes in an XPath array.

The text color of Comment actions can be customized.
New Features

- Where an action's settings includes the selection of a page source, this selection can additionally be specified via an XPath expression.

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

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

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

**Miscellaneous**
- A new Find & Replace Pane enables you to search for strings in the design, including in XPath expressions, functions, and action groups. Found strings can also be replaced.
- A new Listings Pane displays various kinds of lists; for example, lists of all global variables, all user-defined functions, or instances of a particular control type. These listings contain links to the relevant design components, which enables you to quickly find a design component and go to it.
- You can copy a pre-existing style sheet and paste it as a new style sheet.
- MobileTogether Server Services can be configured to send push notifications (PNs) to standard MobileTogether solutions as well as to MobileTogether AppStore Apps.
- The Network Proxy options dialog enables you to configure custom proxy settings for the application.
- The SPL language that is used to generate program code for AppStore Apps for Android, iOS, and Windows has been extended with functionality to trim strings of specified characters from left and right.
- The context menu of the Styles & Properties Pane has been extended with commands to (i) list controls having the same value as that of the selected property, and (ii) display controls in groups according to the values of a given property.
- Page sources that are supplied with data from a FlowForce job can now accept HTML or JSON data, as alternatives to the already supported XML format.
- REST functionality in MobileTogether now supports the sending of files; this is in addition to the sending of XML and Base64 data.
- Multiple font files can be embedded in the solution. These files can be referenced via CSS when the solution is displayed in web clients. The font files to embed are specified in the Browser Settings of the project’s properties.
2.3  Version 4

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

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

Server services
- A server service is a set of MobileTogether Designer actions that is deployed to MobileTogether Server Advanced Edition as a solution (.mtd file). The service is executed on the server when a specified set of MobileTogether Server conditions is met. (These server conditions are defined in the administrator interface of MobileTogether Server Advanced Edition.)
- A server service is defined in a server service design, which is opened via the File | New Service 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 additionally have a Text Size Auto-Fit property, which enables text to be automatically re-sized to fit the width of the control. Controls can also be assigned to a group, so that all controls have an automatically selected uniform and reasonable size. All the controls of a page for which the auto-fit property has been set can be listed in the Listings pane by using the Page menu command List Text Size Auto-Fit Groups.

• 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 address); (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 the 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 page 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.
2.4 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.
New Features

- 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.
2.5 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
New Features

**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 page 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.
2.6 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 `SMT_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 a project's page 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.
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](#)

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\ |
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 7 SP1 with Platform Update, Windows 8,
### MobileTogether Server

<table>
<thead>
<tr>
<th>Windows</th>
<th>Windows 7 SP1 with Platform Update, Windows 8, Windows 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server</td>
<td>Windows Server 2008 R2 SP1 with Platform Update or newer</td>
</tr>
<tr>
<td>Linux</td>
<td>- CentOS 7 or newer</td>
</tr>
<tr>
<td></td>
<td>- RedHat 7 or newer</td>
</tr>
<tr>
<td></td>
<td>- Debian 8 or newer</td>
</tr>
<tr>
<td></td>
<td>- Ubuntu 16.04 LTS or newer</td>
</tr>
<tr>
<td>macOS</td>
<td>macOS 10.13 or newer</td>
</tr>
</tbody>
</table>

### MobileTogether Client

<table>
<thead>
<tr>
<th>iOS</th>
<th>9 and higher for Apple mobile devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>4.1 and higher for Android mobile devices</td>
</tr>
<tr>
<td>Windows RT, Metro</td>
<td>Windows 10; Windows RT for Windows touch-enabled PCs and tablet computers</td>
</tr>
<tr>
<td>HTML</td>
<td>HTML browsers for any other mobile devices</td>
</tr>
</tbody>
</table>
3.2 Terminology Q&A

How does the MobileTogether system work?

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

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

- An MTD file is a native MobileTogether Designer document.
- Each MTD file contains one MobileTogether project.
- A MobileTogether project consists of one or more pages. 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 page 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 page sources. Such data is accessed using the XPath/XQuery language.
- Via its controls, therefore, a page presents data to the end user and can accept modifications to its data sources.

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

- Control events and their actions: 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 page source. You can add page sources (data sources) to the default page, and you can add more pages to the project (see the points below). Create a new MTD file with the File | New 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 as page sources, and each page source is shown there as a tree of nodes. Data from these nodes is used by the controls in the page design, for display, or for processing that leads to some kind of data representation (such as charts or images). Nodes in page source trees are addressed using XPath expressions. Client data input can also be saved back to the page sources if desired. See the section Page Sources 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 page 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 OnButtonClick, 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 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.
Introduction

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 in MobileTogether Designer. To get you started using a more learning-based approach, see the following:

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

Locator expressions

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

- `/Company/Office`: Locates all Office child elements of the Company element, which is the top-level document node. We know that the Company element is the top-level element because it occurs directly under the root node, which is indicated by the first forward slash.
- `/Company/Office[3]/@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 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.

```xml
if ($MT_iPad=true() ) then "Apple" else ""
```
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\MobileTogetherDesigner7\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\Examples` folder. These examples show how various features of MobileTogether Designer can be used, and so can be used as reference points for your own designs.

**Video demos of how to get started with MobileTogether**

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

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

File paths in Windows 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.

<table>
<thead>
<tr>
<th>Windows 7/8/10</th>
<th>C:\Users&lt;username&gt;\Documents</th>
</tr>
</thead>
</table>

• 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.

<table>
<thead>
<tr>
<th>Windows 7/8/10</th>
<th>C:\Program Files\Altova\</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit version on 64-bit OS</td>
<td>C:\Program Files (x86)\Altova\</td>
</tr>
</tbody>
</table>

**File URLs in example files**
File URLs in example files might be absolute URLs. In such cases, they will probably not correspond to the directory structures of your work environment. If you use the supplied example files, make sure that file URLs in them—and any XPath expressions that build such URLs—point to the correct file locations on your machine or network.
4.1 QuickStart (Part 1)

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

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

- Create a New Design
- Set Up a Page
- Add a Page Source (or 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 QuickStart tutorial folder

Video demo of how to build your first app

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

4.1.1 Create a New Design

In this part, you will:

- Create a new MobileTogether design
- Save the design to file
- Set up the design's preview properties: the preview device and magnification level
- Familiarize yourself with the GUI of MobileTogether Designer
A MobileTogether design is created in an MTD (MobileTogether Design) file, which has the `.mtd` extension and is native to MobileTogether Designer.

To create a new MobileTogether design, do the following:

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

2. The new design will be in a tab called *New Design1*. Click **File | Save**, and save the file with any name to any location you like. Make sure, however, that you save your design file in the same folder as the splash screen images. (Since your design, when complete, will show the splash screens of Altova products, you could, for example, call it `AltovaSplashScreens.mtd`.) The new file name, which will have a `.mtd` extension, replaces *New Design1* in the file's tab. The file name also appears in the application's title bar.

3. In the **Main toolbar of MobileTogether Designer**,** 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.
4.1.2 Set Up a Page

A MobileTogether design can consist of one or more pages. A page is what the end-user sees on the mobile client device. If there are multiple pages in a design, the page sequence is defined by their sequential order in the Pages Pane. 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.

![Pages Pane Screenshot](image)

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 page sources. (No page 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.
4.1.3 Add a Page Source (or Data Source)

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

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

Buttons in this section

Add Source

A page can have one or more page sources from where data can be obtained. Page sources are defined in the Page Sources Pane 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 page sources that can be used are described in the section Page Sources (Data Sources). In this tutorial, we will add one type of page source, an editable empty XML source, for which we will then create a structure and content directly in the Page Sources Pane.

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

1. Click the Add Source button in the toolbar of the Page Sources Pane 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 page 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 page source's tree. The data provided by a default file can be overridden by data that is manually entered in a tree node. In our case, there is a single data node: $XML1/Products/Product$, and it has the string xmlspy as its default content. The XML data structure is:

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

**Set a Page XPath Context Node**

Each page has a Page XPath Context Node, which is used as the context node for relative XPath expressions defined on that page. The Page XPath Context Node is indicated in the Page Sources Pane with the label XPath Context (see screenshot below). Note that each page has its own Page XPath Context Node.
We will change the Page XPath Context Node to the `Products` node. Do this by right-clicking `Products` and selecting `Set As Page XPath Context`. The element `Products` will be set as the Page XPath Context Node—this is indicated by the node being labeled `XPath Context` (see screenshot below).

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

### 4.1.4 Format the Design

The design page now has a name (`SplashScreens`), a display title (`Altova Splash Screens`), and a page 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.

4.1.5 Add a Control: Combo Box

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

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

Buttons in this section

Additional Dialog
Add a combo box

Add the combo box and define its properties as follows:

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

![Altova Splash Screens](image1.png)

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.

![Edit Combo Box](image2.png)

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 page 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.

4.1.6 Add a Control: Image

We will now add an image to the design. This image will be the splash screen of the Altova product that the end user selects in the combo box (see previous section). 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.
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 \textit{Reload} action must be set on the combo box. How to do this is described in the next section, \textit{Define Control Actions}.

### 4.1.7 Define Control Actions

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

\textit{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 \textbf{Page | Actions Overview}. The Page Actions Overview dialog (\textit{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.
4.1.8 Validate the Project

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

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

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

4.1.9 Run a Simulation

You can run a simulation of the project workflow directly within MobileTogether Designer. The simulation device will be the device currently selected in the preview device combo box 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.

![Altova Splash Screens](image)

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

![Page Sources](image)

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).

4.1.10 Deploy to Server

After you have successfully validated the project and tested the simulations, you are ready to deploy the project to MobileTogether Server. When the project is deployed to the server, it becomes a solution that a MobileTogether Client app can run. For more information about deploying the project and files, see the sections Deploying the Project 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 page source. You will learn how to generate a dynamic dropdown list for the combo box from the data in the XML file, how to create dynamic links to website pages, and how to save data back to the XML file.

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

- Load Data from a File
- 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

Video demo of how to build your first app

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

4.2.1 Load Data from a File

In this part, you will:

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

Listing of the XML file, AltovaProducts.xml
Located in the MobileTogether folder of the (My) Documents folder:
MobileTogetherDesignerExamples\Tutorials\AltovaProducts.xml.

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

Specify data file to use on page load

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

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

8. Click **OK** to finish.

**Modify the data structure of the page source**

The XML data file has an extra **Selection** element. So, in order for the XML tree to hold the data from this element, we will now add a **Selection** element to the XML tree of the page source in the **Page Sources Pane** (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.

4.2.2 Change Source Node

In this part, you will:

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

Change source node of the combo box

Each page control can have a source node (or page source link), which is a node in one of the page sources. The link is made by dragging the page source node from the Page Sources Pane 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.
4.2.3 Run a Simulation

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

<table>
<thead>
<tr>
<th>What you see</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Selection</code> node in XML Data tree is empty</td>
<td>Value comes from the empty <code>Selection</code> node in <code>AltovaProducts.xml</code>, the file that is loaded on page open</td>
</tr>
<tr>
<td>Combo box is empty</td>
<td>Because the <code>Selection</code> node is empty</td>
</tr>
<tr>
<td>Dropdown list has empty entry</td>
<td>Empty entry added as a result of current combo box selection (=empty)</td>
</tr>
<tr>
<td>No splash screen is displayed</td>
<td>Image URL is built using the empty <code>Selection</code> node</td>
</tr>
</tbody>
</table>
If you now select a product from the dropdown list (for example, MobileTogether), the splash screen of that product will be displayed (screenshot below).

![Altova Splash Screens](image)

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.

### 4.2.4 Use File Data for Combo Box Entries

In this part, you will:

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

Listing of the XML file, AltovaProducts.xml

Located in MobileTogether folder of 

```
My Documents folder
```

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

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

**4.2.5 Set Data File as Default File**

*In this part, you will:*

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

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

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

![Page Sources](image)

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`.

   ![AltovaProducts.xml](image)
3. On clicking OK, the file will be added as the default file, and its data will populate the page source tree.
4. Click Page | Page Actions to open the Page Actions dialog.
5. In the OnPageLoad tab, select the Load from File entry and delete it. This is because the Load from File action is now redundant since the file AltovaProducts.xml has been specified as the default file of the page's only page source.
6. Run a simulation to test whether the default file is being used.

4.2.6 Create Dynamic Links to Web Pages

In this part, you will:

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

Add a button that links to a web page

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

1. Drag the button control from the Controls Pane 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.
4.2.7 **Save Data Back to File**

*In this part, you will:*

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

**Save data to file after editing combo box**

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

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

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

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

That's it!
4.3 Simple Database

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

![Screenshot of the currently available cars](image)

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

Video demo of how to build a database-driven app

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

4.3.1 The DB Data Source

In this section, a new MobileTogether design is created. It uses the Microsoft Access database MyCars.mdb as its data source.
The structure of the DB is as shown in the screenshot below. (Note that the screenshot shows only the first few records of the DB.)

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Fuel</th>
<th>Cylinder</th>
<th>Horsepower</th>
<th>YearFrom</th>
<th>YearTill</th>
<th>Source</th>
<th>Series</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW</td>
<td>550i xDrive Gran Turismo</td>
<td>Gas</td>
<td>8</td>
<td>445</td>
<td>2013</td>
<td>2014</td>
<td>US 5</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>550i xDrive GranTurismo</td>
<td>Gas</td>
<td>8</td>
<td>400</td>
<td>2010</td>
<td>2012</td>
<td>US 5</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>640i Convertible</td>
<td>Gas</td>
<td>6</td>
<td>300</td>
<td>2012</td>
<td>2014</td>
<td>US 6</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>640i Coupe</td>
<td>Gas</td>
<td>6</td>
<td>300</td>
<td>2012</td>
<td>2014</td>
<td>US 6</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>640i xDrive Coupe</td>
<td>Gas</td>
<td>6</td>
<td>300</td>
<td>2014</td>
<td>2014</td>
<td>US 6</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>645Ci</td>
<td></td>
<td></td>
<td>325</td>
<td>2004</td>
<td>2005</td>
<td>US 6</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>645Ci CONVERTIBLE</td>
<td></td>
<td></td>
<td>325</td>
<td>2004</td>
<td>2005</td>
<td>US 6</td>
<td></td>
</tr>
<tr>
<td>BMW</td>
<td>650Ci</td>
<td></td>
<td></td>
<td>360</td>
<td>2006</td>
<td>2010</td>
<td>US 6</td>
<td></td>
</tr>
</tbody>
</table>

Set up the DB data source in the design

Set up the DB data source as follows:

1. Create a new design (File | New).
2. Create a page source by clicking the Add Source icon in the Page Sources pane.
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 page 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.

### 4.3.2 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:

![Currently Available Cars](image)

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 page 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 page source link. As a result, when the end user selects a dropdown-list item in the combo box, then the selected item's corresponding XML value will be saved to the $PERSISTENT/Root/Manufacturer$ node.

After the end user selects a new dropdown-list item in the combo box, the $DB1$ tree should be reloaded. (This is because the data to load from the DB data source must be selected on the basis of the combo box selection. So when the combo box value changes the data selection must also change. How the DB data is selected on the basis of the combo box selection will become clearer at the end of the next section.) 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.

4.3.3 Load DB Data Based on User Selection

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

Table for displaying records

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

![Currently Available Cars Table](image)

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.

![New Table Control](image)
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 \(\text{label}\) 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 (see screenshot below), and create them as label \(\text{label}\) controls: Manufacturer, Model, Horsepower (see screenshot below).

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

![Table Design](image)

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:

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

- As soon as the user changes the value in the combo box, the Reload action 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.
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
<table>
<thead>
<tr>
<th>Title</th>
<th>The World Is Flat: A Brief History of the Twenty-first Century</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Thomas L. Friedman</td>
</tr>
<tr>
<td>ISBN</td>
<td>0374292795</td>
</tr>
<tr>
<td>Publisher</td>
<td>Farrar, Straus and Giroux (NY)</td>
</tr>
<tr>
<td>Pages</td>
<td>616</td>
</tr>
<tr>
<td>Year</td>
<td>2006</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Moneyball: The Art of Winning an Unfair Game</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Michael Lewis</td>
</tr>
<tr>
<td>ISBN</td>
<td>0393324818</td>
</tr>
<tr>
<td>Publisher</td>
<td>W. W. Norton &amp; Company</td>
</tr>
<tr>
<td>Pages</td>
<td>317</td>
</tr>
<tr>
<td>Year</td>
<td>2004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Market Volatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Robert J. Shiller</td>
</tr>
<tr>
<td>ISBN</td>
<td>0262691515</td>
</tr>
<tr>
<td>Publisher</td>
<td>Mit Press</td>
</tr>
<tr>
<td>Pages</td>
<td>480</td>
</tr>
<tr>
<td>Year</td>
<td>1992</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>The Art of War</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Sun Tzu</td>
</tr>
<tr>
<td>ISBN</td>
<td>1590302257</td>
</tr>
<tr>
<td>Publisher</td>
<td>Shambhala Publications</td>
</tr>
<tr>
<td>Pages</td>
<td>273</td>
</tr>
<tr>
<td>Year</td>
<td>2005</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Triplanetary (Lensman, #1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>E.E. &quot;Doc&quot; Smith</td>
</tr>
<tr>
<td>ISBN</td>
<td>1882988093</td>
</tr>
<tr>
<td>Publisher</td>
<td>Old Earth Books</td>
</tr>
<tr>
<td>Pages</td>
<td>287</td>
</tr>
<tr>
<td>Year</td>
<td>1997</td>
</tr>
</tbody>
</table>
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

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

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

4.4.1 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.
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).

### 4.4.2 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 the 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.

4.4.3 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).
4.4.4 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.
4.4.5 Adding Records

To add a record, click the Add Record button of the top page (see screenshot below).
<table>
<thead>
<tr>
<th>Title</th>
<th>The World Is Flat: A Brief History of the Twenty-first Century</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Thomas L. Friedman</td>
</tr>
<tr>
<td>ISBN</td>
<td>0374292795</td>
</tr>
<tr>
<td>Publisher</td>
<td>Farrar, Straus and Giroux (NY)</td>
</tr>
<tr>
<td>Pages</td>
<td>616</td>
</tr>
<tr>
<td>Year</td>
<td>2006</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Moneyball: The Art of Winning an Unfair Game</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Michael Lewis</td>
</tr>
<tr>
<td>ISBN</td>
<td>0393324818</td>
</tr>
<tr>
<td>Publisher</td>
<td>W. W. Norton &amp; Company</td>
</tr>
<tr>
<td>Pages</td>
<td>317</td>
</tr>
<tr>
<td>Year</td>
<td>2004</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Market Volatility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Robert J. Shiller</td>
</tr>
<tr>
<td>ISBN</td>
<td>0262691515</td>
</tr>
<tr>
<td>Publisher</td>
<td>Mit Press</td>
</tr>
<tr>
<td>Pages</td>
<td>480</td>
</tr>
<tr>
<td>Year</td>
<td>1992</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>The Art of War</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>Sun Tzu</td>
</tr>
<tr>
<td>ISBN</td>
<td>1590302257</td>
</tr>
<tr>
<td>Publisher</td>
<td>Shambhala Publications</td>
</tr>
<tr>
<td>Pages</td>
<td>273</td>
</tr>
<tr>
<td>Year</td>
<td>2005</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Title</th>
<th>Triplanetary (Lensman, #1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>E.E. &quot;Doc&quot; Smith</td>
</tr>
<tr>
<td>ISBN</td>
<td>1882968093</td>
</tr>
<tr>
<td>Publisher</td>
<td>Old Earth Books</td>
</tr>
<tr>
<td>Pages</td>
<td>287</td>
</tr>
<tr>
<td>Year</td>
<td>1997</td>
</tr>
</tbody>
</table>
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 page sources so that DB table data can be displayed and edited
- How to display DB data based on end user choices
- How to create charts that are based on the DB data

The screenshot below shows the first page of the DBAndCharts.mtd solution. The end user can select the office and year for which sales are to be displayed. These selections are made in combo boxes at the top of the design. The total sales for that year are then displayed in the Licenses column of the results table below the combo boxes. Whenever a new office or year is selected in either of the combo boxes, 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 page sources of the design.

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

4.5.1 The Project Structure
Open DBAndCharts.mtd and validate it (Project | Validate) to check that the file correctly connects to the Access DB, OfficeSales_DB.mdb. If there is a connection error, make sure you correct this before proceeding. (See the section, Page Sources of the Main Page, 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.
4.5.2 The Main Page

The design of the *DB Sales Main Page* is show below. Its components are numbered in the callouts and are described below.
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 page sources, the appropriate source node has been assigned by dragging the page source node onto the control. The combo boxes and buttons additionally have actions associated with their events. Actions are assigned in the Actions dialog for the control, which is accessed by right-clicking the control and selecting the Control Actions for... command.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A label control to display the title of the page; style properties applied</td>
</tr>
<tr>
<td>2</td>
<td>Combo boxes for end user selection of Office and Year. See detailed description</td>
</tr>
<tr>
<td>3</td>
<td>A horizontal line control as layout component; style properties applied</td>
</tr>
<tr>
<td>4</td>
<td>Table control with cells that contain DB data. See detailed description</td>
</tr>
<tr>
<td>5</td>
<td>Chart controls that display DB data in the form of charts. See detailed description</td>
</tr>
<tr>
<td>6</td>
<td>Button controls with OnButtonClicked actions that go to Edit Offices and Edit Sales pages</td>
</tr>
</tbody>
</table>

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

### 4.5.3 Page Sources of the Main Page

The main page has three page sources: $XML1, $DB1, and $DB2. These are displayed and managed in the Page Sources Pane (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 page source: $XML1
This page source was created as an editable empty XML. The root node $XML1 contains a root element (root), which has two attributes (DesiredOffice and DesiredYear). The root node, $XML1, was set (via its context menu) as the XPath context node for the page two. No default file is set, so no data is imported into the tree.

This page 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 on Load (XPath Value)). This is because the value of the node appears in the associated combo box, and we want the combo box to have an initial selection (see simulator screenshot above). The XPath expressions that provide the initial values are:

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

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

Additionally, we have specified that the @DesiredOffice node is correctly filled whenever the main page loads. This is done with an Update Node action on the OnPageLoad event of the main page (Page | Page Actions).
The action updates the @DesiredOffice node. If this is the first loading of the page, then the ID of the first office is passed as the content of @DesiredOffice. Otherwise the value is what is already present in @DesiredOffice. The result of this is that during an execution, the value in @DesiredOffice is not changed, but the value is initialized whenever the page is loaded for the first time.

The second page source: $DB1

The second page source ($DB1) is the Offices table in the MS Access database, *OfficeSales_DB.mdb*. The data for this page source 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 on Load (XPath Value)**:

```xquery
let $all := $DB1/DB/RowSet/Row/@id
let $ids := remove($all, index-of($all, ""))
let $id := if (empty($ids)) then 1 else max($ids) + 1
return $id
```
Note that the id value is the unique ID number of the office, whereas the City value is the name of the city in which the office is. This is important because while it is the id that is used to uniquely identify an office (via the $XML1/root/@DesiredOffice node), it is the name of the city that we use to identify an office to the end user.

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

The third page source: $DB2

The third page source ($DB2) is the Sales table in the MS Access database, OfficeSales_DB1.mdb. 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 page source tree. The table's columns correspond to the attributes of the Row element. The id attribute has an XQuery expression to generate the id value for every new row that is created. The XQuery expression is inserted by using the context menu command, Ensure Exists before Page Load (XPath Value):

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

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

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

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: \( \text{min}(\text{distinct-values}\{\$DB2/DB/RowSet/Row[@Office=$XML1/root/@DesiredOffice]/@Year\}) \). 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.

4.5.5 The Tabular Report

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

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.)
  
  \[
  \text{sum}(\text{$DB2/DB/RowSet/Row[@Office=\$XML1/root/@DesiredOffice][@Year=\$XML1/root/@DesiredYear]/@Licenses$})
  \]

### 4.5.6 The Charts

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

```xml
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**.

![Change Appearance dialog](image)

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 $DB1$.
  \[
  \text{for } \$i \text{ in distinct-values(Row/@Office) return } \$DB1/DB/RowSet/Row[@id=\$i]/@City
  \]

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

### 4.5.7 Edit Offices Table

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

```xquery
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:

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

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 \texttt{@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:

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

### 4.5.8 Edit Sales Table

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

<table>
<thead>
<tr>
<th>Office</th>
<th>Year</th>
<th>Licenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vienna</td>
<td>2010</td>
<td>700</td>
</tr>
</tbody>
</table>

#### Vienna by Year

- **2010**: 700 (4%, 2010)
- **2011**: 3091 (17%, 2011)
- **2012**: 3100 (17%, 2012)
- **2013**: 6790 (37%, 2013)
- **2014**: (4830, 26%)

[Edit Offices Table] [Edit Sales Table]
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):**

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

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).

![New Table Dialog](image)

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

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

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

The tutorial files
The files for this tutorial are located in your (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
- Page source Listings
- Top Page: Page Sources
- Top Page: Customers Table
- Top Page: Action Group, Go to Sub Page
- Top Page: Show All Orders Action
- Sub Page: Page Sources
4.6.1  Design Structure

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

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

Design steps

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

**Top page: Customers**

- Create the top page and two page sources: $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 page 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
Create the three page sources for the sub page:\n
- $XML1$ (shared with top page), $CUSTOMERS$ (shared with top page), and $ORDERS$
- Create a dynamic table to display the order details in the data file\n- Set up the visibility property of the table's repeating row group\n- Create an XPath expression to generate the total amount

### 4.6.2 Page Source Listings

The design will have three XML page sources:

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

Listing of the XML page source, Customers.xml

Located in the MobileTogether folder of the (My) Documents folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<Customers>
Listing of the XML page source, Orders.xml

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

```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>
  </Order>
</Orders>
```
4.6.3 Top Page: Page Sources

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

1. Create a new design file with the **File | New** command.
2. Save the file with any name you like.
3. In the **Pages Pane**️️, 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 page source called **$XML1**️️ will be created (**see screenshot below**️️).
5. Manually build the structure of this page source so that it is as shown in the screenshot below. Do this by right-clicking nodes in the tree (starting with **$XML1**) and using the **Add Child**, **Append**, and **Insert** context menu commands.
6. Right-click the **CustomerCode**️️ node, and select the command **Ensure Exists on Load (fixed value)**️️. 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 page 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 page 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).
4.6.4 Top Page: Customers Table

We will now create a table to display details of all the customers currently stored in the XML page source Customers.xml. 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 page 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 page source onto the label (see screenshot below). This label will display the contents of the customer's Name element.

7. Create the contents of the other table columns similarly: by dragging the City, ZipCode, and Country elements onto the respective labels (see screenshot below).

8. Select all four labels (by pressing Ctrl while selecting each label), and apply whatever label formatting you like (via the Styles & Properties Pane).
4.6.5 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 page source. In the sub page, the selected-customer data is used in the Visible property of the table row group to filter the orders. We need therefore to do the following when the end user makes a selection by clicking a customer-detail label:

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

These are the actions that will need to be performed when any of the labels in a customer row is clicked. Since the same sequence of actions must be executed for each label, we can save the sequence of actions in a common action group, 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 page 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).

![Manage Group Actions](image)

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.

![Action Group: Show Orders](image)

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.

4.6.6 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.

### 4.6.7  Sub Page: Page 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 page sources:

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

**Adding the page sources**

Add the three page sources as follows:

1. In the Page Sources Pane, 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 page source, $CUSTOMERS, in the same way. Note that the structure, content, and default file will be as created in the top page.

4. In the Page Sources Pane, 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 page source called $XML2 will be created.

5. Double-click the root node $XML2 and edit the name to $ORDERS (see screenshot below).

6. Click the Additional Dialog button of the $ORDERS default file. 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 page sources have been added, you are ready to create the design of the sub page.
4.6.8  Sub Page: Orders Table

We will now create a table to display the orders of the customer that is selected on the top page by the end user. The orders are stored in the XML page source `Orders.xml`. 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 page 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.

<table>
<thead>
<tr>
<th>Customer</th>
<th>Order</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>789: JuniorsRV</td>
<td>002/2015-04-02</td>
<td>EUR 8948.60</td>
</tr>
<tr>
<td>789: JuniorsRV</td>
<td>003/2015-04-06</td>
<td>EUR 2786.45</td>
</tr>
</tbody>
</table>

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 its XPath icon. (Alternatively, you can: (i) right-click the Text property and select Calculate with XPath, or (ii) select the Text property and click the XPath icon in the pane's menu bar.) In the XPath dialog that appears, enter the respective XPath expressions. Note that the XPath context node is the respective $ORDERS/Orders/Order element.

For the Customer column

```xml
  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

```xml
  concat(@number, '/', OrderDate)
```

For the Amount column

```xml
  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.

### 4.6.9 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 page 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 page 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.

### 4.6.10  Sub Page: Decimal Totals in XPath

To complete the design, we will add a label that displays the total amount of the displayed orders. Do this as follows:

1. Drag a **Label control** 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 **Edit XPath/XQuery Expression dialog** that appears, enter the XPath expression to calculate the total amounts (the expression is given below), and click **OK**.

```
if ($XML1/Root/CustomerCode!='All') then concat('Total: ',sum($XML1/Root/Order/Amount))
```
The XPath expression to calculate the total amount

We need to calculate totals in two events: (i) for the orders of the selected customer, and (ii) for all orders. This can be done with the following XPath expression:

```
if ($XML1/Root/CustomerCode!='All')
then concat('Total: ', xs:decimal(sum ($ORDERS//Order[CustomerCode=$XML1/Root/CustomerCode]/OrderAmount)))
else concat('Total: ', xs:decimal(sum ($ORDERS//OrderAmount)))
```

This XPath expression works as follows:

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

The selected OrderAmount elements are summed using the `sum()` function of XPath. Since the `sum()` function uses the `xs:double` type and returns an `xs:double` number, the amount will have more than the two decimal places we require in a currency. We therefore use the `xs:decimal` type converter to round the `xs:double` to a two-decimal place number.

### 4.6.11 Simulation and Testing

After completing the design, run a simulation (by pressing F5) and test your design. The screenshots below show the top page and sub pages in the MobileTogether Designer simulator.

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

<table>
<thead>
<tr>
<th>Customer</th>
<th>Order</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>789: JuniorsRV</td>
<td>002/2015-04-03</td>
<td>EUR 8345.50</td>
</tr>
<tr>
<td>789: JuniorsRV</td>
<td>005/2015-04-06</td>
<td>EUR 2786.45</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td><strong>11132.05</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
<th>Order</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>456: HiDeHo</td>
<td>001/2015-04-03</td>
<td>EUR 4906.38</td>
</tr>
<tr>
<td>789: JuniorsRV</td>
<td>002/2015-04-03</td>
<td>EUR 8345.50</td>
</tr>
<tr>
<td>123: New Fashion</td>
<td>003/2015-04-04</td>
<td>EUR 5645.20</td>
</tr>
<tr>
<td>123: New Fashion</td>
<td>004/2015-04-05</td>
<td>EUR 3805.58</td>
</tr>
<tr>
<td>789: JuniorsRV</td>
<td>005/2015-04-06</td>
<td>EUR 2786.45</td>
</tr>
<tr>
<td>456: HiDeHo</td>
<td>006/2015-04-07</td>
<td>EUR 7460.50</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td><strong>32949.81</strong></td>
</tr>
</tbody>
</table>

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 page sources are used. The $PERSISTENT tree is used to hold the customer database. A $EDIT tree is used to hold the one record (a Customer element) that is being currently edited.
- The $EDIT/Customer element is created when an Add New button is clicked.
- When a Save button is clicked, the $EDIT/Customer element is appended as the last element of the $PERSISTENT tree. The record is then deleted from the $EDIT tree (so that a record can be loaded subsequently, either new or for editing).
- The customer database is displayed as a table that shows the Customer elements in the $PERSISTENT tree (see screenshot above).
- When any field of a record in the customer database is clicked, then that record is loaded in the $EDIT tree, where it can be edited.
- When the Save button is clicked after a record has been edited in this way, it is saved back to the $PERSISTENT tree at its original location.

The tutorial file

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

4.7.1 Design Pages

When the solution starts, the main (top) page, named All Addresses, is displayed. This page contains: (i) a table displaying all the records in the customer database (contained in the $PERSISTENT tree), and (ii) an Add New button that enables the user to add new records to the customer database.
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)*.
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).

**4.7.3 Add a New Record**

When the user opens the solution, the customer database is empty. To add a record, the Add New button (see screenshot below) is clicked.
The **Add New** button has two **onButtonClicked** actions ([Reload](#) 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.

### 4.7.4 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.

### 4.7.5 Display All Records

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

- The table has a single header row that is located outside the dynamic row.
- The dynamic row—that is, the row that repeats—is linked to the $PERSISTENT/Root/Customer page source element. As a result, in the solution, a new row is created for each Customer element.
- The cell of each table column contains a label control 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 `$EDIT/Customer`. Remember that this page source contains the **one customer record that is currently being edited**.
- The deleted attribute nodes of the `$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.
4.7.6 Edit an Existing Record

When the end user clicks a field in a customer record, the solution goes to the *Edit Address* sub page (screenshot below), where the user can edit that specific record.

- 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*[^1]. 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 page 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.

### 4.8.1 The XML Page Source

We need one XML page source ($XML$) 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 source CityTimes.xml, showing the document's structure**

```xml
  <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>
    </City>
  </Cities>
</CityTime>
```

© 2015-2021 Altova GmbH
<TimeZone>UTC+8</TimeZone>
<Time>12:00 AM</Time>
</City>
...
</Cities>

Full listing of the XML page source, CityTimes.xml
Located in the following MobileTogether folder of the (My Documents folder): MobileTogetherDesignerExamples\Tutorials\SoapRequests.

<?xml version="1.0" encoding="UTF-8"?>
<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>
Adding the XML page source

An XML page source (XML) that uses CityTimes.xml as its page 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:

```xml
```

- 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.

### 4.8.2 Design Components

The design components are numbered in the screenshots below and are described in callouts further below. The screenshot on the left shows the simulation, the screenshot on the 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 SMT_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).

![Screenshot of the page design](image)

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.

4.8.3 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.

### 4.9.1 Reading and Sharing the Geolocation

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

![My Geolocation](image)

- 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 as a page source 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.

### 4.9.2 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).

![My Geolocation](image)

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:

```xml
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**.
Scrollable Tables

- **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.

### 4.10.1 Tables that Force Full Screen Height

The first page of the design, *Use Full Page Height*, contains a table with dynamic rows 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 Max Table Height property is set to Rest of screen height (always). This ensures that additional space is added below the table so that the last component of the page is displayed just above the bottom of the screen.
• The table's Scroll Vertically property is set to Whole table. This ensures that the footer is kept with the body of the table (see screenshot below). Otherwise, the footer would be positioned just above the rest of the page's content, which might lead to an unsightly gap between the last row of the table and the footer.

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

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

4.10.2 Tables Having Specific Heights

The second page of the design, Table at 50 Percent, contains a table with dynamic rows that is created with the $XML2/Customers/Customer element as its repeating element (see screenshot below). This table is similar to the table created on the previous page. The difference is that while the previous table auto-adjusted its height so that all page components filled the full screen height, this table is defined to have a height that is 50% of the screen height (see the table's Max Table Height property in the screenshot below).
As a result of the 50% Max Table Height setting, the two buttons that have been created below the table will be positioned—in the output—directly below the 50% table (see screenshot below). We have also set the table’s Scroll Vertically property to Rows except header and footer. This ensures that the header and footer are kept fixed in the fixed-height table, while the body rows scroll (see screenshot below).

Note: In the MobileTogether Designer Simulator, use the scroll wheel to scroll vertically, and click-and-drag to scroll horizontally.
You can modify the values of properties to test the various possibilities. See the section Table Properties for details of scrollable table properties.
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
- Breakpoints 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.
5.1.1 Page Design

The Page Design View (Page View for short) is the view in which the page that is going to be deployed to the mobile device is designed (see screenshot below).

To design a page, drag-and-drop controls from the Controls Pane 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 and device settings
The screenshot below shows the Page View settings in the Main toolbar. The settings you make here also determine the settings of the device used in simulations.

- **Preview Device**: This combo box allows you to select various mobile devices, so that the design can be previewed for specific devices. The selected device will be not only the device used in the design, but also that used in simulations. You can change the preview device at any time.
- **Portrait/Landscape Preview Toggle**: Toggles the design preview between portrait and landscape.
- **Zoom level**: A combo box to select the zoom level in 10% steps from 10% to 100%. The zoom level can also be modified via the **View 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.

5.1.2 DB Query
The DB Query View (screenshot below) enables you to directly query any major database from within the MobileTogether Designer GUI. The database could be a page source referenced in the active document or an external database. Note that each DB Query pane is associated with the currently active design. You can have connections to multiple databases for a single design. There can also be multiple designs open in MobileTogether Designer. Queries and actions defined in the DB Query View are independent of other MobileTogether Designer tabs, and are not saved as part of the .mtd design file.
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, tabbed pages, and control templates

There are four kinds of page-type components at the project level:

- **Top pages**: A top page is part of the sequence of pages that makes up a workflow. When the solution is started, it progresses from the first page to the last, in the order listed in the Pages Pane. You can change the position of a page in the list by dragging the page to a new position. Tabbed pages are also part of the workflow sequence, but sub pages are not.

- **Tabbed pages (or tab splits)**: A tabbed page (or tab split) is a page with tabs, each of which contains a page. For example, in the screenshot above, the tabbed page (indicated by Tab Split) is defined to have two tabs that contain, respectively, the pages Sales US and Sales EU. Tabbed pages are part of the page sequence that makes up the project's workflow.

- **Sub pages**: A sub page is not part of the sequence of pages that make up the workflow of the project. It is similar to a module that is called by a control in a top page (or tabbed page). For example, an OnButtonClick event of a Button control in a top page could use the GoToSubpage action to go to a particular subpage, and then return to the top page (or go to another page). You can declare parameters and variables for a sub page by clicking its Add Parameters/Variables button.

- **Control templates**: A control template resembles a page in that you can lay out a set of controls on it. It is called a template because it can be reused (with all its controls) on other pages. You can declare
parameters and variables for a control template by clicking its **Add Parameters/Variables** button. See [Control Templates (CTs)](#) for more information.

Also see **Pages, Tabbed Pages, and Sub Pages** for more information about these kinds of pages.

### Adding, renaming, and deleting pages
Besides the methods listed in the following table, you can also use context menu commands for some of these tasks *(see further below)*.

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a page</td>
<td>Click the <strong>Add Page</strong> icon in the pane's toolbar. From the dropdown menu, select <strong>Add Top Page</strong>, <strong>Add Tab Split</strong>, or <strong>Add Sub Page</strong>. A new page is added to the Pages Pane with a name of <em>New Page X</em> or <em>Process Tab Split</em>, and the empty new page is displayed in the Main Window.</td>
</tr>
<tr>
<td>Rename a page</td>
<td>Double-click the name in the Pages Pane and edit the name.</td>
</tr>
<tr>
<td>Delete a page</td>
<td>Select the page you want to delete and click the <strong>Delete</strong> icon in the pane's toolbar, or press Delete.</td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a page before a top page, sub page, tabbed page, or child page of a tabbed page</td>
<td>Right-click the page and select <strong>Insert Page</strong></td>
</tr>
<tr>
<td>Add a top page at the end of the <em>Top Pages</em> list</td>
<td>Right-click any item and select <strong>Add Top Page</strong></td>
</tr>
<tr>
<td>Add a sub page at the end of the <em>Sub Pages</em> list</td>
<td>Right-click any item and select <strong>Add Sub Page</strong></td>
</tr>
<tr>
<td>Add a child page at the end of a list of child pages of a tabbed page</td>
<td>Right-click the tabbed page and select <strong>Add Page as Child</strong></td>
</tr>
<tr>
<td>Add a tabbed page before the selected page</td>
<td>Right-click and select <strong>Insert Tab Split</strong></td>
</tr>
<tr>
<td>Add a tabbed page after the last page of the <em>Top Pages</em> list</td>
<td>Right-click in either list and select <strong>Add Tab Split</strong></td>
</tr>
<tr>
<td>Add a control template before the selected page</td>
<td>Right-click and select <strong>Insert Control Template</strong></td>
</tr>
<tr>
<td>Add a control template after the last control template</td>
<td>Right-click in either list and select <strong>Add Control Template</strong></td>
</tr>
<tr>
<td>Find references in the design to the selected page</td>
<td>Right-click the page and select <strong>List Usages</strong></td>
</tr>
</tbody>
</table>
5.3 Files Pane

The **Files Pane** *(screenshot below)* enables you to add files that you want to deploy to the server and/or client. Some files, such as the default files of page sources, are added automatically to the list of deployable files. But you can also add files directly via the context menu (obtained by right-clicking inside the Files Pane). The deployable files that you add here are read-only and can be image files and other files, such as XML files. They are organized in the pane under two headings: **Image Files** and **Other Files** *(see screenshot below)*.

![Files Pane Screenshot](image.png)

Each file has a check box next to it, indicating whether it is deployed. To select whether to deploy a file to the server and/or client, right-click it and select the appropriate **Deploy on...** command. To not deploy a file, deselect 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)*.
- Copy the path of a selected file via the corresponding command in the pane's context menu.
- List all usages of the selected file via the corresponding command in the pane's context menu. This shows where in the design the deployed file is invoked and helps you to quickly locate the usage/s.
- 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.
- Where the file will be deployed (server and/or client) is displayed below each file name. This can be changed via the corresponding context menu command.
- Check the box in front of each file name to deploy the file. 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, then do not deploy it but store it on the server at a location that the MTD file correctly references. *(See the sections **Location of Project Files** and **Deploying the Project** for more information.)* Deploying a file subsequently with another design does not affect previously deployments.

+ Deploy only if the file is used as a read-only file.
+ Deploy if, for some reason, the file's URL will not be accessible to clients.
Adding and removing deployable files from within the Files Pane

<table>
<thead>
<tr>
<th>To...</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a file</td>
<td>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.</td>
</tr>
<tr>
<td>Remove a file</td>
<td>Right-click the file you want to remove and, from the context menu that appears, click Remove File.</td>
</tr>
</tbody>
</table>

Other ways to add deployable files to the Files Pane
Files can also be added to the Files Pane in the following ways:

- When a file is added as a page source or when an image file is added to a page design, a dialog is displayed that asks whether the file should be deployed to MobileTogether Server. If you click Yes, the file is added to the Files Pane with its check box selected. If you click No, the file is added to the Files Pane with its check box unselected. You can select/deselect a file's check box in the Files Pane (see screenshot above) at any later time.
- In the context menu of the root node of a page source 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).
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.
In the context menu of a control (obtained by right-clicking the control), select the **List All** command to display in the **Listings Pane** all instances of that control type that occur in the active design.
5.5 Breakpoints Pane

The Breakpoints Pane (screenshot below) lists all the project's breakpoints and tracepoints and provides a convenient single place to manage these debug points. Breakpoints can be set on actions and XPath expressions, and are indicated by a solid red circle. Tracepoints can be set on XPath expressions, and are indicated by a solid blue circle. Both types of debug points may be disabled, in which case they are ignored during debugging sessions; disabled debug points are indicated by a solid gray circle with a border color that indicates which type of debug point it is (red for breakpoints, blue for tracepoints).

The following features are available:

- To filter what types of debug points are shown, right-click in any free area of the pane.
- To enable/disable individual debug points: (i) click its circle indicator, or (ii) right-click the debug point and toggle the **Enabled** command on/off.
- To enable/disable all debug points, click the respective toolbar icon.
- To remove individual debug points: (i) right-click the debug point and select **Delete**, or (ii) select the debug point and click the toolbar icon **Clear This Debug Point**.
- To remove all debug points, click the toolbar icon **Clear All Debug Points**.
- You can enter a break condition, which is an XPath expression that must evaluate to `true()` for the breakpoint to be enabled. To set a break condition for a given breakpoint, double-click **Enter break condition** in that breakpoint's right-hand column. Alternatively, right-click the breakpoint and select **Edit**. The second breakpoint in the screenshot above has a break condition. Note that break conditions are available for breakpoints only, not for tracepoints.
5.6 Page Sources Pane

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

The pane provides the following main functionality:

- Displays all the page sources of the page currently selected in the Pages Pane
- Displays all the namespaces declared for the active project
- Enables page sources to be added to the page, via the toolbar's Add Source icon
- Enables namespaces to be added to the project, via the toolbar's Add Namespace icon
- Enables the default data file of a page source to be set. The data file provides the data that goes into the nodes of the page source
- Enables elements and attributes to be added to a tree, relative to the selected node
- Enables elements and attributes to be given fixed values or XPath-generated values on page load
- Enables the default XPath context node to be set (the highlighted node in the screenshot above). The selected node will be the context node for all XPath expressions defined for that page
- Enables nodes to be associated with controls in the page design. This is done by dragging the node onto the control. The associated node (called the page source link) is displayed in bold. When you hover over such a node in the page source tree, a popup provides information about the associated
control/s in the design. Controls that are associated with a page source link have an icon at the control's top left. Hovering over the icon displays information about the associated page source link.
- Enables items in the pane (namespaces, trees, elements, and attributes) to be deleted

For detailed information about how to manage and work with page sources, see the section, Page Sources (Data Sources).

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 page source, and click the appropriate toolbar command (see toolbar screenshot below). Temporary elements and attributes are intended to hold data used for calculations or data that for any other reason should not be saved to file. The data of temporary nodes is not saved.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Command</th>
<th>Does this...</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="add-source-icon" alt="Add Source" /></td>
<td>Add Source</td>
<td>Displays the Add Page Source dialog. A root node is created for the page source that is added. Only one child element can be added to a root node.</td>
</tr>
<tr>
<td><img src="add-namespace-icon" alt="Add Namespace" /></td>
<td>Add Namespace</td>
<td>Inserts or appends a namespace declaration under the Namespace entry. Edit the default prefix if you want, and enter a namespace.</td>
</tr>
<tr>
<td><img src="add-element-icon" alt="Add Element" /></td>
<td>Add Element</td>
<td>Inserts, appends, or adds a child element relative to the selected node.</td>
</tr>
<tr>
<td><img src="add-attribute-icon" alt="Add Attribute" /></td>
<td>Add Attribute</td>
<td>Inserts, appends, or adds a child attribute relative to the selected node.</td>
</tr>
<tr>
<td><img src="delete-icon" alt="Delete" /></td>
<td>Delete</td>
<td>Deletes the selected node.</td>
</tr>
</tbody>
</table>
5.7 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.
5.8 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 Pane

#### 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**: 

© 2015-2021 Altova GmbH
Styles & Properties Pane toolbar

The commands available in the Styles & Properties toolbar (screenshot below) are listed in the table below:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Command</th>
<th>Does this...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>List Non-Empty</td>
<td>Toggles between displaying all properties and only those that have values defined for them. Properties that have default values are considered empty.</td>
</tr>
<tr>
<td></td>
<td>Expand All</td>
<td>Expands all sub-panes.</td>
</tr>
<tr>
<td></td>
<td>Collapse All</td>
<td>Collapses all sub-panes.</td>
</tr>
<tr>
<td></td>
<td>Edit XPath</td>
<td>Enabled when a property that requires an XPath expression is selected. It displays the XPath/XQuery Window.</td>
</tr>
<tr>
<td></td>
<td>Reset</td>
<td>Resets the property to empty or to its default value.</td>
</tr>
</tbody>
</table>

Entering and editing property values

Property values are entered or edited depending on the ways that are available to enter the value. These ways are listed in the table below. Often, more than one way is available to enter a value, enabling you to choose whatever way you prefer. If it is possible to enter a value by way of an XPath expression, then the property will have an XPath icon in the third column, or the XPath icon in the toolbar will be enabled when the property is selected.

<table>
<thead>
<tr>
<th>Entry method</th>
<th>Do this...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text field</td>
<td>Double-click, and enter or edit the property value.</td>
</tr>
<tr>
<td>Combo box</td>
<td>Select a value from the dropdown list. Alternatively, enter or edit a value.</td>
</tr>
<tr>
<td>Color palette</td>
<td>Click to pop up the color palette, then select a color.</td>
</tr>
<tr>
<td>Edit XPath</td>
<td>Click to pop up the Edit XPath/XQuery Expression dialog, then enter an XPath/XQuery expression. The expression provides the value of the property.</td>
</tr>
<tr>
<td>Additional dialog</td>
<td>Click to open an additional dialog that will enable a value to be entered. This dialog could be, for example, the Actions dialog or the Open File dialog.</td>
</tr>
</tbody>
</table>

Context menu

The context menu of a property provides a quick way to access the additional functionality listed below.

- Set a fixed value from a list of valid pre-selected values.
- Set an XPath expression that will, at run time, evaluate to the value of the property.
- Set a platform default. This option is enabled if the default value is variable across platforms.
- Reset to the default value of the property. The default value of a property can be seen if you place the cursor over the name of the property.
- Cut, Copy, Paste commands.
- Access the MobileTogether Designer Style Sheets dialog, in which styles can be managed at various levels (from document-wide to a single control type).
- List, in the Listings Pane, controls that have the same direct style value as that of the selected property. A direct style value refers to a value that is entered directly at the control level—as opposed to a value that is set via a style sheet.
- List, in the Listings Pane, controls that have the same style value as that of the selected property, whether the value is entered directly at the control level or is set via a style sheet.
- Group, in the Listings Pane, controls by their direct style value. All controls that use the selected property will be listed, organized into groups according to the value that has been set for the selected property. For example, if this command is executed on the Text Size property, then controls are listed in groups for different text sizes (large, medium, 10pt, etc). Direct style value refers to a value that is entered directly at the control level—as opposed to a value that is set via a style sheet.
- Group, in the Listings Pane, controls by their style value, whether the value is entered directly at the control level or is set via a style sheet. The effect is similar to that of the previous command; it groups controls on the basis of property values.
5.9 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.

Using XPath trace() to generate messages

You can use XPath's `trace()` function to generate messages in the Messages Pane. For example, the following XPath expression will generate a message that reflects the value in the `Company/Name` node:

```xml
if (Company/Name="Altova") then trace("Company is Altova") else trace("Company is not Altova")
```

The text of the relevant `trace` function is returned as the message in the Messages Pane. This message will also be a link, which when clicked takes you to the the originating `trace` function.

Filtering messages

You can specify what kind of messages are displayed in the Messages Pane. To do this, click the Filter
button in the toolbar of the Messages Pane (screenshot above). This displays the Filter Settings dialog (screenshot below). Select the message types you want to display and click Close. This feature can be very useful, for example, if there are too many messages and you wish to focus on just one type of message.

*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.
5.10 Listings Pane

The Listings Pane (screenshot below) displays the output of listing and grouping commands, such as menu command Project | List Usages of All Global Variables.

Note the following points:

- The pane has nine tabs, so you can leave the results in one tab and switch to a new tab. This can be useful, for example, if you want to compare lists.
- The window is not cleared each time a new list is displayed, but each list is displayed in a new color (see screenshot above).
- Lists and their components can be expanded and collapsed.
- Components in the list are hyperlinked. So you can immediately go to a component, or to the dialog where a component is defined, by clicking that component in the list.
- The toolbar of the Listings Pane contains commands that enable the following: navigation of the lists, copying of list items, searching the lists, and clearing the tab.

Available listing and grouping commands
Lists generated by the following commands are displayed in the Listings Pane.

- List All Controls Of Kind
- Group Controls By Style Value
- Group Controls By Direct Style Value
- List Controls With Same Style Value
- List Controls With Same Direct Style Value
- List Usages of All Global Variables
- List Usages of All Page Source Variables
- List Page Sources by Attribute
- List Usages of All User-Defined XPath/XQuery Functions
- List Usages of All Action Groups
- List Usages of All Stylesheets
- List All File and Directory References
- List All External Data References
- List Unused Functions, Variables, Etc
- List Text Size Auto Fit Groups
- List All Usages of an Action Kind
- List All Page Usages
5.11 Find & Replace Pane

The Find & Replace Pane (screenshot below) enables you to search for text strings in all pages of design or in the current page. The search scope includes XPath expressions and XPath function names, variable names and values, control names and properties, and actions and action groups. You can also replace all instances of the found text string and use regular expression to find text.

Searching for text

To search for text in the design, do the following:

1. In the Find What entry field (see screenshot above), enter the string to find, or use the dropdown list to select a string from one of the last 10 strings. You can also use regular expressions (explained below) to find text.

2. Optionally, narrow or expand the search by, respectively, selecting or deselecting the Match whole word and/or Match case options. If Match whole word is selected, then only the exact words in the text will be matched. For example, if the search term is fit, then only the word fit will be matched; the fit in fitness, for example, will not be matched.

3. Select the scope of the search by toggling on/off the options in the Find Options combo box. You can restrict the search to the current page, and include XPath functions and/or action groups in the search.

4. Click Find or press Enter.

Results of the search

There are nine Result tabs (see screenshot below). Results are displayed in the currently selected Result tab (and overwrite any previous results in that tab). You can switch to a new tab if you want to keep a previous result.
Result tabs provide the following functionality:

- Each instance of a found search term is displayed on a separate line and contains links to the relevant design objects. For example, in the screenshot above, the instance that is highlighted refers to an instance of the search term Selection in an XPath expression that is defined for an Open URL action of a Button control. Clicking the three links take you, respectively, from left, to the action definition, the button control in the design, the XPath expression of the action.

- You can use the Copy icons in the toolbar of the Results tab (see screenshot above) to copy the following to the clipboard: (i) the selected result, (ii) the selected result and its children, (iii) all results.

- The Results tab has its own Find function, which enables you to search the results for a term. Click the Find icon in the toolbar to do this. Navigate these results with the Find Previous and Find Next icons.

- The Clear icon of the Results tab toolbar clears all the results in the current tab.

Replacing text
To replace the search term in the found instances with another text string, enter the new string in the Replace With text box and click Replace. Note that a Replace action might not work if the replaced text causes invalidity, for example, when a replacement in an XPath expression invalidates the expression, or when a replaced style value is not a valid value.

Using regular expressions
You can use regular expressions (regex) to find a text string. To do this, first, switch the Regular expression option on (see Searching for text above). This specifies that the text in the search term field is to be evaluated as a regular expression. Next, enter the regular expression in the search term field. For a brief description of regular expression metacharacters, see the section Regular expression metacharacters below.

Regular expression metacharacters
Given below is a list of regular expression metacharacters.

<table>
<thead>
<tr>
<th>Metacharacter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Matches any character. This is a placeholder for a single character.</td>
</tr>
<tr>
<td>(</td>
<td>Marks the start of a tagged expression.</td>
</tr>
<tr>
<td>)</td>
<td>Marks the end of a tagged expression.</td>
</tr>
<tr>
<td>{abc}</td>
<td>The { and } metacharacters mark the start and end of a tagged expression. Tagged</td>
</tr>
</tbody>
</table>
expressions may be useful when you need to tag ("remember") a matched region for the purpose of referring to it later (back-reference). Up to nine expressions can be tagged (and then back-referenced later, either in the Find or Replace field).

For example, \1 matches the string the. This expression can be literally explained as follows: match the string "the" (and remember it as a tagged region), followed by a space character, followed by a back-reference to the tagged region matched previously.

\n Where \n is a variable that can take integer values from 1 through 9. The expression refers to the first through ninth tagged region when replacing. For example, if the find string is Fred([1-9])XXX and the replace string is Sam\1YYY, this means that in the find string there is one tagged expression that is (implicitly) indexed with the number 1; in the replace string, the tagged expression is referenced with \1. If the find-replace command is applied to Fred2XXX, it would generate Sam2YYY.

\< Matches the start of a word.
\> Matches the end of a word.
\x Allows you to use a character \x, which would otherwise have a special meaning. For example, \[ would be interpreted as [ and not as the start of a character set.

[...]] Indicates a set of characters. For example, [abc] means any of the characters a, b or c. You can also use ranges: for example [a-z] for any lower case character.

[^...] The complement of the characters in the set. For example, [^A-Za-z] means any character except an alphabetic character.

^ Matches the start of a line (unless used inside a set, see above).
$ Matches the end of a line. Example: A+$ to find one or more A's at end of line.
* Matches 0 or more times. For example, Sa*m matches Sm, Sam, Saam and so on.
+ Matches 1 or more times. For example, Sa+m matches Sam, Saam, Saaam and so on.

**Representation of special characters**

Note the following expressions.

| \r | Carriage Return (CR). You can use either CR (\r) or LF (\n) to find or create a new line |
| \n | Line Feed (LF). You can use either CR (\r) or LF (\n) to find or create a new line |
| \t | Tab character |
| \ | Use this to escape characters that appear in regex expression, for example: \\

© 2015-2021 Altova GmbH
6 Project

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

- Client-Server Interaction
- Location of Project Files
- Deploying the Project
- MobileTogether Packages
- 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.

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.
6.2 Location of Project Files

The project file (.mtd file) is the solution file. It is deployed to the server and accessed from there by the MobileTogether Client app. When the project file is deployed to the server it is stored in the server's database of solutions, and the server will reference the file by its name. The project file uses other files (such as XML files and image files), from which it reads data and to which it can write data. These associated files can be stored at the following locations:

- **Deployed to the server with the project file**
  - These data files will be read-only on the server.
  - The files are stored in the server's database. The advantage is that access to the data files is internally handled within the project.
  - A file can be referenced in the design with a relative or absolute path. The filepath in the design (relative or absolute) will be used as the internal file reference on the server database.
  - For the details of deployment, see: Deploying the Project, Files Pane, Deploy to MobileTogether Server.

- **Embedded within the project file**
  - This applies to XML data files (typically the default files of page sources).
  - The advantage of embedding is that the XML data and images will be transported together with the project file, and will be correctly accessed from within the project. Therefore, server access is not required in order to access these files.
  - As with deployed files, embedded XML files will be read-only.
  - The disadvantage is that the size of the project file increases.
  - To embed a file, right-click its root node 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 page source, security authorizations can be set when specifying the properties of the page source.
- The advantages of using Internet locations are: (i) data accessed by the solution will always be up-to-date, and (ii) the solution is portable.

Also see
- Deploy to MobileTogether Server
- 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
Access to a solution can be controlled on two levels: (i) accessibility of the server where the solution is deployed; (ii) a client device's access to a server.

Accessibility of the server
- Deployment to in-house servers behind a firewall automatically restricts access to internal users, for example, to the employees of a company.
- Deployment to servers that allow external access allow external end users to access MobileTogether solutions, for example, the customers or clients of a company.

Client access to the server
For each server a set of users can be defined that have access to that server. Access will be available only to those client devices that submit the appropriate user-name and password. The users of a server and their privileges are defined in settings of MobileTogether Server. See the user manual of MobileTogether Server for details of how to define users, roles, and user privileges.

Deployed files and their locations
The following files are deployed when the project is deployed with the File | Deploy to Server command:

- The project (.mtd) file, which is deployed to the server. This is the solution file that will be accessed by the MobileTogether Client app on the client device.
- Deployable files in the Files Pane that have their check boxes selected. These files will be read-only and typically are image files and data files. They can be deployed to the server or client or both. If a file is deployed to the client, then it is transferred from the server to the client once, when the solution is started. If a file is looked up frequently, then it might be better to deploy it to the client. This would save the time required to transfer it from the server each time the file is called.

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 page 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.

Note: If you are deploying a MobileTogether package—and not a MobileTogether solution—then you do not have to select deployable files. All required resources are contained in the package and will be deployed automatically to the server.
Deployed files are read-only

Deployed files are read-only. If there is a Save action defined for a deployed file, then the design will be invalid—since the deployed file is read-only and cannot be written to. Deployed files should be used to display information or visuals (images, charts, etc) or to read data.

If you want to save data to a file or database, then store the file at a location in the Server Side Solution's Working Directory and reference the file correctly in the design. To build a correct reference to the file, (i) specify the file's location when it is added, and (ii) make sure that the Server Side Solution's Working Directory is correctly set in MobileTogether Server. See the section Location of Project Files | A directory on the server for more information.

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.
Deploying the Project

Note: The Automated test runs pane displays the test cases that have been saved to the design. It appears only if at least one test case has been saved to the design.

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 **MobileTogether Packages**

A MobileTogether package is a zip file that is generated from a MobileTogether design. It contains the design in an encrypted form and, optionally, deployable resource files that are used by the design (such as images and CSS stylesheets). The package file has a `.mtp` file extension. Generating a package from the design enables you: (i) to create a package of the design and its related files for easy portability, and (ii) to open the package in MobileTogether Designer and quickly **deploy the design as a solution to MobileTogether Server**.

Note the following points:

- If a solution requires resources that are not deployable and that were, therefore, not included in the package, then these resources must be saved manually to the server. For more information, see [Deploying the Project](#) and [Export MobileTogether Package](#).
- After the package has been generated, do not modify it internally or in any other way. Any modification will make the `.mtp` file unopenable in MobileTogether Designer.
- The `.mtp` file can be opened in MobileTogether Designer only to deploy the design contained in it as a solution to MobileTogether Server. You cannot open the `.mtp` file and then (i) edit the MobileTogether design which is inside the package, or (ii) simulate the design's workflow in MobileTogether Designer.

How to generate a MobileTogether package is explained in the description of the [Export MobileTogether Package](#) command.

To deploy the design that is contained in a MobileTogether package to MobileTogether Server: (i) open the `.mtp` file in MobileTogether Designer, and (ii) **deploy the package just as you deploy a design**.
6.5  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.

### Styles & Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Access</td>
<td>always</td>
</tr>
<tr>
<td>Name</td>
<td>Project</td>
</tr>
<tr>
<td>Timeout Client Waiting</td>
<td>10</td>
</tr>
<tr>
<td>Data Retrieval on Server</td>
<td>OnPushNotificationReceived</td>
</tr>
<tr>
<td>Theme</td>
<td>Enforce Light Theme</td>
</tr>
<tr>
<td>Audio Actions</td>
<td></td>
</tr>
<tr>
<td>Text to Speech Actions</td>
<td></td>
</tr>
<tr>
<td>NFC Actions</td>
<td></td>
</tr>
<tr>
<td>Push Notification</td>
<td>OnPushNotificationReceived</td>
</tr>
<tr>
<td>In-App Purchase Actions</td>
<td>OnPurchaseUpdated</td>
</tr>
<tr>
<td>Ask User on Exit Workflow</td>
<td>true</td>
</tr>
<tr>
<td>Exit Workflow Message</td>
<td>Do you really want to exit this workflow</td>
</tr>
<tr>
<td>On Switch to Other Solution</td>
<td>Cancel this solution</td>
</tr>
<tr>
<td>Workflow Icon</td>
<td></td>
</tr>
<tr>
<td>Browser Settings</td>
<td></td>
</tr>
<tr>
<td>More Project Settings</td>
<td>continue action handling on XPath error, Sequence of Values</td>
</tr>
</tbody>
</table>

#### 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

- 219
- 291
- 1006
This is the amount of time the server waits for data to be retrieved from a source external to the server (from a DB or URL, for example). The value is an integer value in seconds that can be entered or selected from the dropdown list of the combo box. The default value is 10 seconds. If the timeout period is exceeded, then an error message is displayed. An exception to this is when load actions have the setting On error set to Continue. In this case, the On Error actions of that action’s Continue setting are executed.

### Theme

Three options are available: (i) Use System Settings, (ii) Enforce Light Theme, (iii) Enforce Dark Theme. The default is Use System Settings. A light theme is one in which dark text is displayed on a light background. A dark theme is one in which light text is displayed on a dark background. If the style definition is Use System Settings, then the choice of theme is determined by the user's device.

### Audio Actions

Audio events are defined globally for the entire project. Three events are available: OnAudioStarted, OnAudioError and OnAudioCompleted. The actions that are defined for these events apply to all Audio playback events in the project. Clicking the property’s Additional Dialog button displays a dialog containing the definitions of the project's Audio events. For each event, you can define the actions to perform by dragging and dropping actions from the left-hand Actions pane into the event's tab. For more information, see the description of the Audio (Playback) feature.

### 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.

### In-App Purchase Actions

The In-App Purchase Actions button takes you to the Actions dialog for the OnPurchaseUpdated event. During an in-app purchase, after the end user on the client device confirms a purchase, the app store sends data about the purchase to the device. The data will be stored in a new Purchase element of the $MT_IN_APP_PURCHASE page source. It is when this update to the page source occurs that the OnPurchaseUpdated event is triggered. In the Actions dialog of the OnPurchaseUpdated event, you can define actions that you want to perform at this point in the in-app purchase process. Typically, you would (i) check the response code of the purchase transaction for success/failure, and (ii) acknowledge the purchase.

See the topics Purchase Products and Example Project: The "Buy" Button 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. The maximum pixel size of workflow icons is 200x200. By default, the MobileTogether icon will be used.

▼ Browser Settings

Clicking the Browser Settings property's Additional Dialog button displays the Browser Settings dialog (screenshot below). Here you can define certain settings related to the browser of the mobile device. These settings are described below.
The following settings can be defined:

**Desktop Browser Orientation**
The combo box options enable you to select the orientation of the browser: *Force Portrait* and *Force Landscape*. The default is *Force Portrait*.

**CSS File**
For styling in web clients—that is, in browsers—only, this setting specifies the external CSS file that is read in order to evaluate CSS properties assigned to the class selectors of controls in the design. An external CSS file can be modified at any time in order to change the appearance of design components.
Each design component has a property called **Browser CSS Class**, which defines a CSS class name specific to that control. CSS properties for these class selectors can then be defined in an external CSS file, which is deployed to the server. The CSS file to look up for the class rules is specified in this (**CSS File**) setting. You can select the CSS file via a file path or global resource alias. You can also use an XPath expression to generate the file path (see screenshot above). Note the following points: (i) the CSS rules defined in the external CSS file have a lower priority than the definitions that are made in a control's properties; (ii) the CSS file is not available in simulations for web browsers.

**Font File**
Specifies one or more font files that you want to embed in the design and use in addition to the system fonts. You can either browse for the font file, locate it with a global resource, or generate its filepath with an XPath expression. The following font file types are supported: `.ttf`, `.otf`, `.woff`, `.woff2`. Additionally, MobileTogether will correctly generate `.eot`, `.svg`, and `.svgz` fonts; however, these font file types are not supported by all browsers. If you want to embed multiple font files, enter an XPath expression which is a string containing the multiple filepaths separated by commas (see screenshot above). A font that is embedded in the design in this way can be referenced via the `font-family` property of CSS. If a font has been embedded that is also available on the local system, then the system font will be used. If you specify the same font in different file types (say, WOFF2 and TTF), then the browser will download the one file type that it supports best, and none of the alternative file types. For more information about CSS and browser-related font information, see the following MDN web pages: `@font-face` and `font-family`.

**Allow Back Navigation in <iframe>**
This property applies to **Embedded Webpage Solutions**, which are solutions that are loaded into an `<iframe>` element of a webpage. If this option is set to `true`, the Back navigation button of the browser allows the end user to navigate back within the `Iframe`. The default is `false`.

**Default Font Size**
Select the base font size from which all control font sizes are calculated. The **Browser Default** option selects the default size of the browser. The default option is `16px`.

**Confirm Browser Window/Tab Close**
This option enables a message box to be displayed when the end user wants to close the browser window or browser tab that is displaying the solution. The message (i) asks whether the user really wants to leave the page, and (ii) informs the user that if he or she clicks Leave, then unsaved changes might not be saved. This setting applies to all pages in the project. If this option is selected, and you want to disable it for one or more pages individually, then you must set the **Browser CSS Class** property of each of these pages (see **Page Properties**) to a value of `mt-no-browser-exit-confirmation`.

**Fonts for Rich Text Control**
Add the fonts you want to allow the end user to select from. These fonts will be displayed in the dropdown list of the font selection combo box of the **Rich Text Control**. If no font is specified in this list, then the font selection combo box will be disabled in the solution.

**Note**
Relative file paths that are set in this dialog are relative to: (i) the solution directory on the server, and (ii) in MobileTogether Designer, to the directory in which the design is located.

**More Project Settings**
Clicking the **More Project Settings** property's Additional Dialog button displays the More Project Settings dialog (screenshot below). Note that settings are divided across two tabs.
More Project Settings

Settings 1

Re-Deployment Timeout
Duration (in hours) a deployed design remains in the server database after a new version has been re-deployed. Up until this time clients using this design can finish their workflow.

XPath Compatibility Mode
Enable XPath compatibility for XQuery statements

Abort action handling on errors
Abort action handling when an XPath or other error occurs.

Ignore Default Namespace in HTML Documents
Remove the 'xmlns=' assignment in HTML documents to simplify XPath statements.

Automatic Re-embedding
Automatically re-embed page sources when deploying or simulating the design.

Save device visualization
Save device type, zoom, and device orientation along with the design and use it upon loading.

Settings 2
The following settings can be defined:

- **Re-deployment Timeout**: The time in hours after a new version of a solution is deployed that the superseded solution is kept on the server. This overlap time enables clients currently using the older solution to complete work. The default is 5 hours.

- **XPath Compatibility Mode**: When set to `true`, XQuery constructs that are invalid in XPath are resolved so that XQuery statements containing these constructs are compatible with XPath and can be used where XPath expressions are allowed. Currently, this concerns XQuery entity and character references, which are allowed in XQuery, but not in XPath. When **XPath compatibility mode** is set to `true`, XQuery entity and character references are read in XPath as text; they are not resolved. The default value for this setting is `true`.

- **Abort Action Handling on Errors**: Aborts the handling of actions when an error occurs. The error could be in an XPath expression or elsewhere in the action handling. However, minor errors such as XPath errors for selecting a style property are ignored and action handling continues. The default value is `true`. 
• **Ignore Default Namespace in HTML Documents:** Since only one default namespace is allowed in an XML document, not ignoring the default namespace in HTML documents could create errors in reading XML data sources. The default is true: The HTML default namespace is ignored.

• **Automatic Re-embedding:** `Embedding` refers to the embedding of page sources in the project (design) file. If Automatic Re-embedding is enabled (true), then page sources are re-embedded when deploying or simulating, ensuring that the latest data source files are embedded and that the data, therefore, is up-to-date. The default is true.

• **Save Device Visualization:** If selected, then the `device settings` (device type, zoom level, and page orientation) will be saved with the design. The design will be re-opened with the last-saved device settings. The default is false.

• **Input Parameters:** Specifies the data structure type of the `$MT_InputParameters` variable. The options are: (i) `Named Parameters`, which is a map data structure (for example: `{"name":"Altova", "location":"Boston"}`) and (ii) `Sequence of Values`, which is a sequence data structure (for example: `("Altova", "Boston")`). The default is `Named Parameters`. For more information, see the description of `$MT_InputParameters`.

• **All Styles:** This setting specifies whether the All Styles property of a design component or page is available or not. The setting's values are true or false, with false being the default. If set to true, then the All Styles property will be displayed in the `Styles & Properties Pane`. The property enables you to set all the styles of the selected component or page conveniently at one location, via an XPath map expression. For information about how to use the All Styles property, see its description, for example, on the `Button` control.

• **Advanced UpdateDisplay Options:** Specifies whether the Update Display action offers advanced options (true) or not (false). The advanced options enable you to specify which controls of a page are updated. If advanced options are not switched on, then all controls of a page are updated. The default setting is false. For details about the Update Display action's options, see `Update Display`.

• **iOS Table Padding:** Specifies whether the standard iOS table padding is applied to tables on iOS devices. The default value of this setting is true. When the setting is true, all tables in the project are given a padding of 9px on right and left, and 5px on top and bottom. If values are also set for any `Padding` property of individual tables, then that `Padding` value (top, right, bottom, or left) is added to the respective iOS table padding value. If iOS Table Padding is set to false, then the default iOS table padding will not be applied, and only the padding values you set for an individual table will be applied. The iOS Table Padding setting enables you to apply or remove the iOS table padding for iOS tables in the entire project.

• **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.

- **Client Session Timeout Message:** You can enter a message to be displayed on the client device when the server has timed out. The message can be either entered directly or it can be obtained as the result of evaluating an XPath expression.
6.6 Localization

A solution is created in a default language. But the text strings used in the solution can be localized (translated) into multiple languages. When the solution runs in a mobile device, the language of the solution is automatically selected to be the same as that of the mobile device. If the solution has not been localized into the language of the mobile device, the default language of the solution is used (see screenshot below).

The localized strings are defined in the Localization dialog (screenshot below) by adding a column for each new language and defining the localized strings in this column. For details, see the description of the menu command, Project | Localization. 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')`.

![Localization dialog](image)

<table>
<thead>
<tr>
<th>Page</th>
<th>Control</th>
<th>Property</th>
<th>Default</th>
<th>de</th>
<th>es</th>
<th>ja</th>
<th>fr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro</td>
<td>Check Box1</td>
<td>Text</td>
<td>Skip this intro in the future</td>
<td>Einführung in Zukunft überspringen</td>
<td>En adelante saltar la introducción</td>
<td>回避在未来跳过</td>
<td>このインタラクションを今後スキップする</td>
</tr>
<tr>
<td>Intro</td>
<td>Intro</td>
<td>Page Title</td>
<td>Intro</td>
<td>Intro</td>
<td>Intro</td>
<td>Intro</td>
<td>イントロ</td>
</tr>
<tr>
<td>Intro</td>
<td>Label1</td>
<td>Text</td>
<td>Expense Reports Demo</td>
<td>Spesenbericht-Demo</td>
<td>Demuestra de los informes de gastos</td>
<td>返金の報告書のデモ</td>
<td>費用の報告書のデモ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>de</th>
<th>es</th>
<th>ja</th>
<th>fr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting Summary department N</td>
<td>Show summary of %1 %2 department report(s) in accounting system</td>
<td>Zusammenfassung von %1 %2 Abteilungsbericht(en) im Buchhaltungssystem anzeigen</td>
<td>Mostrar la suma del %1 %2 departamento(s) en el sistema contable</td>
<td>会計システム内の%1%2部門の報告書の要約を表示する</td>
<td>会計システム内の%1%2部門の報告書の要約を表示する</td>
</tr>
<tr>
<td>Accounts Receivable</td>
<td>Accounts Receivable</td>
<td>Buchhaltung</td>
<td>Cuentas por cobrar</td>
<td>会計</td>
<td>会計</td>
</tr>
</tbody>
</table>

![Export... Import... OK Cancel](image)
Localized solutions can be tested by selecting the simulation language you want via the **Project | Simulation Language** command and then running a simulation.
6.7 Namespaces

Namespaces are important for correctly identifying nodes, and for correctly locating nodes with the use of XPath expressions. The Namespaces item in the Page Sources Pane (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.
6.8 Global Resources

Global Resources in MobileTogether allow you to easily specify different database server names, file locations, or other configuration-specific parameters in a way that lets you easily switch from a development to a production system—and you can do that independently for each mobile solution. Because of this, Global Resources enable you to design and test quickly, and, thus, to save time.

As you develop a mobile solution in MobileTogether Designer, the Global Resources dialog allows you to identify each file, database connection, or directory with a user-defined name that can then be used to reference that resource anywhere in your solution. With Global Resources you can set up different configurations and easily switch your mobile solution modes by selecting a different Global Resources configuration.

Once you deploy a new mobile solution to MobileTogether Server, you can upload this configuration with your solution to control the access of your solution to specific servers even after it has been deployed. Administrators can configure the mobile solution's resource configuration on the fly, so a mobile solution can switch from a testing to a production environment with one click.

For more information about how to work with Global Resources, see the section, Altova Global Resources.
6.9 Performance

You can optimize performance of your MobileTogether solutions by being aware of how MobileTogether Server and the MobileTogether Client app work together, how MobileTogether Server can be used to do the most data-intensive processing and store large amounts of data, and how settings to optimize performance can be made in the project design in MobileTogether Designer. This section describes important optimization concepts.

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

### 6.9.1 Embed XML in Design File

Instead of the solution referencing external XML data sources, any XML data can be embedded directly into the design file. This option is perfect for smaller data sets that are needed on the client side, such as a list of choices for a combo-box or other static data. This data is then transmitted to the client as a part of the overall design file and is always instantly available on the client side every time you run the app. So no additional data transfers between client and server are needed.

To embed a data source in the design file, right-click the data source and select **Embed XML in Design File**.

### 6.9.2 Data Querying with XQuery 3.1

Using the powerful XQuery 3.1 language, you can write expressions that significantly reduce the amount of data being transferred between the server and client. Database views, queries, or web service calls to external data sources will yield raw data for a mobile application. This often contains redundancies or may not be the ideal structure for the intended data display in the mobile application. XQuery’s powerful FLWOR expressions allow you to easily restructure and regroup the data to ensure the most efficient data transfer from server to client and the most useful presentation in the client application.

This new version of XQuery has several new features, including support for maps, arrays, and data in the JSON format.

You can use the **Edit XPath/XQuery Expression Dialog** to create and check XQuery expressions.

### 6.9.3 Data Storage on Servers

The speed with which data for solutions is processed can be enhanced by storing certain types of data on the server:
Data that is used to generate charts and graphs, which are images, does not need to be sent to the client; only the image needs to be sent. So the data used to generate the charts and graphs can be kept on the server, and does not need to be transferred.

**Powerful “Keep Data on Server” Setting**

To reduce the amount of data transmitted over the mobile data network—which improves the performance of any mobile solution—MobileTogether lets you select exactly which data you want to transmit to the client devices and which data to keep on the server. For example, if a certain data set is only necessary to display a graph, then that data can be kept on the server. The graph image will be rendered by the server and transmitted to the client without the underlying data being transferred over the mobile network. For large data sets this produces a significant performance boost.

This setting (to keep data on the server and not send it to the client) is made for a data source at the time the data source is added. The setting is in the Data Retention pane (screenshot below) of the **Add Page Source** dialog. It is also available as a command in the **context menus of root nodes** in the **Page Sources Pane**.

![Data retention](image)

**Note:** For Chart data, do not select the option to keep the chart data on the client only (**Client only**). This will lead to an error.

**Caching**

Using settings in MobileTogether Designer and MobileTogether Server, you can specify caching behavior for all data sources. This boosts the speed of MobileTogether greatly because when the server receives a request from the Mobile App, it will already have the data available. There are two main reasons to create caches: (i) If a page data source generates reports slowly (for example, a large database); (ii) If a data source is not modified often. In such cases, execution of a solution would be faster if data is taken from data caches on the server. In order to keep caches up-to-date, the frequency of cache updates can be specified when the cache is created.

Once a cache has been defined in MobileTogether Designer, it can be used by the data sources of different designs, providing the underlying data structure is compatible.

As pioneered in Altova MapForce and FlowForce Servers, MobileTogether contains more than just the usual caching parameters such as expiry and refresh time. You can manually determine the amount of time that passes before caching again. Also you can define how many unique combinations of multiple query parameters (either for databases or for web services) should automatically be cached. A client requesting the data will now immediately get it from the cache, whereas the server will retrieve it only if the cache time has elapsed. This is beyond simple caching as MobileTogether actually automatically executes the query to whatever interval the
designer specifies. When it is a query with parameters, the designer can specify how many unique combinations of parameters should be cached, and then the server will follow those instructions.

A new cache is defined in MobileTogether Designer for a data source. Right-click a data source in the Page Sources Pane, select Cache Settings, and specify the properties of the cache.

If a data source is defined as having a cache, the cached data will be used when the solution is run. Caches can be used as soon as the solution has been deployed to the server.

### 6.9.4 Persistent Data Storage on Clients

For user-entered data and data that doesn’t change too frequently, you can choose to store data persistently on each client device. This reduces the amount of data being transferred between the server and client, and so increases performance speed. Performance is additionally boosted because the round-trip time between server and client is reduced—even for different sessions of the same user that are hours apart. Persistent data can be defined in the following ways:

- **Default persistent trees**: 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.
7 Page Sources (Data Sources)

A MobileTogether project (or design) consists of one or more pages. Each page can have a set of page sources, which provide the structure of the structured data that the page will use, as well as, optionally, the actual data that the page will use.

Because page sources are so closely linked with providing data, they are also sometimes called data sources. However, note the following distinction in terminology: The term data source is used when referring to the physical entity (like a file or DB) that contains the physical data; the term page source is used to cover (i) the general concept of page sources, and (ii) the source of data within the design.

The structure and data of a page's page sources (see screenshot below left) are available to controls placed on that page—and that page only. However, a single page source can be reused among multiple pages, enabling the source's data to be accessed on multiple pages of the project. To reuse a page source on another page, create it there as a page source that reuses an existing structure (see screenshot below right).
Aspects of page sources
Each page source can be conceptualized as having two aspects:

- **Structure**: The structure of a page source is represented in the form of a tree consisting of element nodes and attribute nodes. The nodes of such a tree can be addressed by using XPath expressions. Data within a node or set of nodes can be: (i) retrieved for display on the page, and (ii) updated (say, by an end user) and placed back inside a node. Both of these operations are possible because the page source is structured and can, as a result, be accessed via XPath.

- **Data**: This refers to the content of the nodes of the page source. The data of all page source nodes is thus available for use in the design. The data in a page source can be obtained from a file. Alternatively, you can assign values to individual page source nodes directly in the Page Sources Pane. This data can be displayed on the page, modified by the end user (on the client device), or processed in specific ways by the design, and then saved back to the page source.

Number and types and of page sources
Each page can have multiple page sources of one or more kinds. For example, the page represented in the screenshot above left has three page sources: one XML page source and two DB page sources.

The following types of page source can be set up for a page:

- XML Sources
- HTML Sources
- JSON Sources
- HTTP Sources
- DB Sources
- XQuery Sources
- FlowForce Jobs

Each type of page source is described in the section **Types of Page Sources**.

Organization of this section
This section is organized as follows:

- Types of Data Sources: describes how to create various types of page sources
- Page Source Properties: describes the properties of page sources
- Page Source Trees: shows how the structure of a source tree and data in the tree's nodes are used in a page
- Caches: describes how the mechanism for caching data on the server works
- Context Menu: describes the context menu commands of the Page Sources Pane
7.1 Types of Page Sources: Adding

You can add the following types of page sources, each of which is described in the sub-sections of this section:

- XML Sources
- HTML Sources
- HTTP Sources (HTTP/FTP, REST, and SOAP)
- DBL Sources
- XQuery Sources
- FlowForce Jobs

After you select the type of page source, you could import the structure of a specific file of that type and, if wanted, data from a specific file. After that you can define the properties of the page source (such as when data from an associated data source file is loaded). You can subsequently, depending on the type of page source, modify the structure, the associated data file, and the properties of a page source, by selecting the appropriate command in the context menu of the page source.

Reusing existing structures

After a page source has been created for one page of a design (top page or sub page), it is available for reuse in other pages of the design. If such a page is available, then the Reuse option is enabled.

The available page sources are listed by the names of their root nodes in the dropdown list of the options's combo box (see screenshot above). Select the page source you want to reuse and click Finish. A new root node is created with the same name and structure as the reused page source (see screenshot below). The number of pages with which the page source is shared is listed (see screenshot below) and the name/s of the sharing pages are displayed when you place your mouse over the root node's name in the tree. You can subsequently change the data structure to that of another page source by selecting another reusable page source in the combo box next to the name of the root node (see screenshot below).
How to add data to the tree (including by assigning a default file) is described in the section, Tree Data. How to modify the tree structure is described in the section Tree Structure.

Page source 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 page source. You must specify, in the next screen of the Add Page Source dialog, whether the page source is an XML, HTML, or JSON type.

The structure of the XML/HTML/JSON file is imported as the structure of the page source (see screenshot below). The structure of an HTML or JSON page source is imported as an XML tree structure. An imported JSON structure will have a root element named json. The XML/HTML/JSON (page source) file is also automatically defined as the default file of the page source. This means that data from the file is used as the data of nodes of the new page source. If the file is selected with a URL, you can use the HTTP or FTP protocol to retrieve the file. The path of the default file can also be specified with an XPath expression; this allows the dynamic composition of file paths, for example, paths that are based on node content in other page sources.
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 page source tree in the Page Sources Pane. For example:

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

will be corrected to:

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

### 7.1.1 XML Sources

If you want to add a new XML page source, click the **Add Source** toolbar button. This brings up the first screen of the Add Page Source dialog (screenshot below). You now have two options:

- To create the structure manually yourself, select **New, empty XML, HTML or JSON**.
- To import the structure from a file, select **New XML, HTML or JSON structure imported from file**. If the
XPath-based file name option is selected, then the Edit XPath/XQuery Expression dialog appears and you can build an XPath expression to generate the file URL you need. Otherwise, a dialog box appears in which you can select the file that provides the structure of the page source. You can browse for the file, or use a file URL or global resource.

Click Next to go to the second screen of the dialog. Here, you (i) specify that the data type of the page source must be XML and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

When you click Finish, a root node named $xml is created for the new page source (see screenshot below). You can change the name of the root node if you like by double-clicking in it to edit. If you specified that the structure of the page source must be imported from a file, then, on clicking Finish, you will be prompted to select an XML file. The page source $xml will, in this case, be created with the structure of the selected file.
You can now (i) create or modify the structure of the page source via the toolbar commands, and (ii) add data to nodes of the page source. How to do this is described in the section, Tree Data.

**Note:** You can change the data type of the page source (to HTML or JSON) via the root node's context menu command Data Type.

### Working with serialized XML

When dealing with data transfers, you might need to work with serialize XML. MobileTogether’s Load/Save String action provides serialization functionality geared to use in its page sources. Serialization of JSON data is discussed in JSON Sources.

#### 7.1.2 HTML Sources

If you want to add a new HTML page source, click the Add Source toolbar button. This brings up the first screen of the Add Page Source dialog (screenshot below). You now have two options:

- To create the structure manually yourself, select New, empty XML, HTML or JSON.
- To import the structure from a file, select New XML, HTML or JSON structure imported from file. If the XPath-based file name option is selected, then the Edit XPath/XQuery Expression dialog appears and you can build an XPath expression to generate the file URL you need. Otherwise, a dialog box appears in which you can select the file that provides the structure of the page source. You can browse for the file, or use a file URL or global resource.
Click **Next** to go to the second screen of the dialog. Here, you (i) specify that the data type of the page source must be HTML and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

When you click **Finish**, a root node named **$HTML** is created for the new page source (see screenshot below). You can change the name of the root node if you like by double-clicking in it to edit. If you specified that the structure of the page source must be imported from a file, then, on clicking **Finish**, you will be prompted to select an HTML file. The page source **$HTML** will, in this case, be created with the structure of the selected file.
You can now (i) create or modify the structure of the page source via the toolbar commands, and (ii) add data to nodes of the page source. How to do this is described in the section, Tree Data.

Note that HTML retrieval is done using a correcting parser. As a result, if an imported HTML structure has an invalid data object model because of missing elements (according to the HTML 5 specification), then these missing elements are added to the page source tree in the Page Sources Pane. For example:

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

will be corrected to:

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

Note: You can change the data type of the page source (to XML or JSON) via the root node's context menu command Data Type.

### 7.1.3 JSON Sources

If you want to add a new JSON page source, click the Add Source toolbar button. This brings up the first screen of the Add Page Source dialog (screenshot below). You now have two options:

- To create the structure manually yourself, select New, empty XML, HTML or JSON.
- To import the structure from a file, select New XML, HTML or JSON structure imported from file. If the XPath-based file name option is selected, then the Edit XPath/XQuery Expression dialog appears and you can build an XPath expression to generate the file URL you need. Otherwise, a dialog box appears in which you can select the file that provides the structure of the page source. You can browse for the file, or use a file URL or global resource.
Click **Next** to go to the second screen of the dialog. Here, you (i) specify that the data type of the page source must be JSON and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

When you click **Finish**, a root node named $JSON is created for the new page source (see screenshot below). You can change the name of the root node if you like by double-clicking in it to edit. If you specified that the structure of the page source must be imported from a file, then, on clicking **Finish**, you will be prompted to select a JSON file. The page source $JSON will, in this case, be created with the structure of the selected file.
You can now (i) create or modify the structure of the page source via the toolbar commands, and (ii) add data to nodes of the page source. How to do this is described in the section, Tree Data.

Note: You can change the data type of the page source (to XML or HTML) via the root node's context menu command Data Type.

Functions to convert JSON to/from string

MobileTogether Designer has two application-specific XPath extension functions to (i) save JSON data to a string, and (ii) load a JSON structure from a string. This is useful, if during solution execution, you want to import or export JSON data, or if you want to obtain the JSON data in a structured form that you can then address with XPath expressions. The two functions are:

- mt-save-json-to-string(): The function takes a JSON node as its single argument, and returns the contents of the node as a string that is serialized in JSON notation. The submitted JSON node could be an entire JSON document, or a part of a JSON document. See MobileTogether Extension Functions for details.
- mt-load-json-from-string(): The function takes a string that is a serialized JSON structure as its argument, converts it to XML, wraps it in an element named json, and returns the whole as a document node. Since a document node is returned, you can append an XPath expression to the function if you want to load a part of the JSON structure. See MobileTogether Extension Functions for details.

Also see the description of the Load/Save String action for more serialization options.

7.1.4 HTTP Sources

This section describes the settings needed to make HTTP/FTP, REST, and SOAP requests. These requests are made either to load data from external sources or to save data to external sources. The requests are made in the following situations:

- When adding page sources: 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.

Adding page sources

A page source containing data can be added via an HTTP or FTP request and the structure added subsequently. After you select this option you can specify whether the page source will be obtained using HTTP/FTP, REST, or SOAP (see screenshot below). If you select HTTP/FTP or REST, you must specify, in the next screen of the Add Page Source dialog, whether the page 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 page 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 page source is loaded. The tree structure, however, is not created. It can be imported and/or created manually. How to create the tree structure is described in the section Tree Structure.

7.1.4.1 Via HTTP/FTP Requests

The settings for HTTP/FTP requests are defined in the Edit Web Access Settings dialog (screenshot below). Enter the request kind, the URL of the target resource, the data format of the target resource (XML, HTML, or JSON), user authentication information, and, optionally, query parameters and headers. In the screenshot below, for example, the GET request is composed using an XPath expression: It targets a .rss page on the http://www.ndbc.noaa.gov website. The name of the RSS page is taken from the /NDBC/buoy node, and the
target page will be parsed as XML. Query parameters and headers can be added to the request. The `charset` header, however, is automatically generated by MobileTogether Designer and will not be overwritten by a `charset` header that you enter in this dialog.

On clicking **OK**, the request is carried out.

Note that HTML retrieval is done using a correcting parser. As a result, if an imported HTML structure has an invalid data object model because of missing elements (according to the HTML 5 specification), then these missing elements are added to the page source tree in the Page Sources Pane. For example:

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

will be corrected to:

```
<table>
  <tbody>
  </tbody>
</table>
```
7.1.4.2 Via REST Requests

The settings for REST requests are defined in the RESTful API Request dialog (screenshot below). This dialog is accessed in two situations:

- When defining a page source that uses a REST request to connect to a data source
- Via the Execute REST Request action; in this case the response to the request can be stored in a $MT_HTTPExecute_Result variable.

You can choose to (i) define your own settings, (ii) import a URL, or (iii) use a WADL file. If you choose to define your own settings, you can specify your own definitions for the individual settings. If you import a URL or use a WADL file, some settings will be defined in the URL or WADL file, and so cannot be modified by you.
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&qws_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.

**$MT_HTTPExecute_Result**
You can choose whether the result should be stored in the $MT_HTTPExecute_Result or not. If stored, you can use the result, via this variable, at other locations in the design. Note that this option is not displayed in the RESTful API Request dialog when the page source is being defined for the first time. It is displayed when the dialog is opened via the Execute REST Request action.

**Request method**
In the Own Settings and URL templates, you must specify the Request method (GET, PUT, POST, DELETE, or some other verb, which we call a custom verb). A custom verb would be one that is used by the server from which you are requesting the data source and the one that you want to use for the request. To use a custom verb, select its option and enter the custom verb in the text field that appears. You can, instead of directly entering the custom verb, use an XPath expression to generate the custom verb.
In the WADL template, the request method is determined by the selection you make in the Choose a Method for a Resource dialog (see above for screenshot), and is automatically selected in the template; as a result these radio buttons are disabled in this template.

**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.

![Authentication Settings](image)

**OAuth authentication**

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

The OAuth system works broadly as follows:

1. At the web service, create an OAuth key (or ID) and secret. Together these are known as an OAuth consumer.
2. Make a note of the web service's OAuth endpoints. There are three endpoints for OAuth 1.0 (Initial Endpoint, Authorization Endpoint, and Token Endpoint), and two endpoints for OAuth 2.0 (Authorization Endpoint and Token Endpoint). The endpoints are usually constant for all consumers.
3. Set up the application to access the web service with these five (OAuth 1.0) or four (OAuth 2.0) pieces of authentication information.

After you have obtained your key and secret from the web service, and noted the endpoints you need, you are ready to set up MobileTogether Designer to access the web service. Do this as follows:

1. In the Authentication Settings dialog (screenshot above), click **Setup OAuth 1.0** or **Setup OAuth 2.0**. The OAuth Settings dialog (screenshot below) appears.

![OAuth Settings screenshot](image)

- **Callback Address**: Set to `http://localhost:8083`. This address is fixed; it is the address of the machine on which you are working.
- **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.
- **Initial Endpoint**: Enter the endpoint declared by the web service. These are usually constant across the web service for all consumers of the service.
- **Authorization Endpoint** and **Token Endpoint**: Enter your key (or ID) and secret.
- **ID/Key** and **Secret**: Click **Authorize** to finish.

2. **Parameters**

   Parameters can be freely added, edited and deleted in the **Define Own Settings** template. 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 prefixed 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 prefixed 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 prefixed 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.

![Parameters Table Screenshot]

**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.

Send content as body
Enter the content's value and the content's MIME type. The content can be entered directly or can be accessed from an XML node.

Send contents as multipart
Enter the name for the content part being sent, the type of content access, the content itself, and the MIME type of the content. The content's value is the actual content being sent. In the screenshot above, the content is obtained, via XPath expressions, from page source nodes. Images are sent in Base64 format or as a file.

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

MobileTogether Designer enables you to make SOAP requests via WSDL. A WSDL file describes what operations are provided by a given web service. The SOAP protocol is then used to call one of these operations (over HTTP). The procedure in MobileTogether Designer for making the request is as follows:

1. Add a page source 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.
### 7.1.5 DB Sources

This option enables you to create a page source structure from a database (DB) and to add data from that database. In the first screen of the Add Page Source dialog *(see screenshot below)*, select *New DB structure*.

Click **Next** to go to the second screen of the dialog, where you can define the properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

On selecting this option and clicking **Finish**, the database (DB) Connection Wizard appears. After making the connection to the DB, you can select the table data to be imported for page source structure and data content. A root node (named $DB by default) is inserted, with the database name next to the name of the root node. You can change the name of the root node if you like by double-clicking in it to edit. The root node also has the DB structure below it. See the [Databases](#) 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.

7.1.6 XQuery Sources

To add an XQuery page source, do the following: In the first screen of the Add Page Source dialog (see screenshot below), select *New XQuery Tree*.

Click **Next** to go to the second screen of the dialog, where you can define the properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

On clicking **Finish**, the *Edit XPath/XQuery Expression dialog* will be opened. Enter an XQuery statement that generates the required data structure (and optionally data), and click **OK**. A page source with a root node...
named $xq$ is created that has the structure specified in the XQuery statement. Right-click this root node, select the **Load Data** command and set the option to **On First Use** or **On Every Page**, as required.

For example, the following XQuery statement would generate the tree shown in the screenshot of the simulation further below.

```xml
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: $xq/\text{weather/\text{location/city}}[\text{\(@\text{name}\) = \text{London}}]$). 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 page source's **Load Data** option that you selected (**On First Use**, **On Every Page**, or **Not Automatically**).
You can add a page source that receives data from an Altova FlowForce job and takes its structure either from a file or a structure you create manually in the Page Sources Pane. To create such a page source, do the following: In the first screen of the Add Page Source dialog (see screenshot below), select New FlowForce job.
Click **Next** to go to the second screen of the dialog, where you can define the properties of the new page source. Here, you (i) specify the data type of the page source (XML, HTML, or JSON) and (ii) define the other properties of the new page source. If you are not sure about how to define these properties, then use the default settings. You can always change the settings later by right-clicking the root node of the page source.

On clicking **Finish**, the Edit FlowForce Settings dialog (screenshot below) opens. Enter the FlowForce settings to connect to a FlowForce server and service. If you enter a URL and then click the **Refresh** button of the Job name entry, a connection to the FlowForce Server will be made and the available jobs will be displayed. When you select a job, its parameters will be displayed and you can enter the desired parameter values. If, however, you wish to compose the FlowForce Server URL using an XPath expression, you must specify the job name in the URL and, in the **Parameters** pane, manually enter the parameters and their values.
On clicking **OK**, a new page source is created that contains data retrieved by running the FlowForce job. This data can be parsed as XML, HTML, or JSON, depending on the selection you made when defining the properties of the page source in the second screen of the Add Page Source dialog. The structure of the page source can be modified subsequently. How to do this is described in the section, **Tree Structure**. For information about Altova's FlowForce application, see the FlowForce page on the Altova website and the FlowForce product documentation.
7.2 Page Source Properties

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

<table>
<thead>
<tr>
<th>Data type</th>
<th>XML</th>
<th>HTML</th>
<th>JSON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data modification</td>
<td>Editable</td>
<td>Read-only</td>
<td></td>
</tr>
<tr>
<td>Data retention</td>
<td>Shared between Client and Server</td>
<td>Client only</td>
<td>Server only</td>
</tr>
<tr>
<td></td>
<td>data will be automatically transferred between Server and Client on program start, tree load/save, modification, etc.</td>
<td>data can only be created on Mobile Client side with action commands and will never be transferred to/from Server</td>
<td>data, usually loaded from databases, server files or web services, will remain for Server processing only</td>
</tr>
<tr>
<td></td>
<td>Persist Data on Client</td>
<td>data will be automatically saved on Mobile Client file system on program end and will be reloaded on the next program start</td>
<td></td>
</tr>
<tr>
<td>Load data</td>
<td>On first use</td>
<td>On every page</td>
<td>Not automatically (load with page action)</td>
</tr>
<tr>
<td>Save Data</td>
<td>On every page leave</td>
<td>On any solution finish</td>
<td>On last submit</td>
</tr>
</tbody>
</table>
Note that some options will not be available for some types of page source. All the available options are listed below.

**Data type**
The format of data in the page source being added. Select from XML, HTML, or JSON. After the page source is added, you can change the data type in the context menu of the page source node (context command Data Type). 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. Note that chart data must not be saved on the client only; doing so will result in an error.

The Persit Data on Client option loads a solution with the client data it had when the solution was last closed. The client data is said to "persist" between two solution runs.

**Load data**
Selects whether data is loaded the first time the page is loaded, every time the page is loaded, or when specified by a page action. The selected option can be changed subsequently via the context menu of the source tree's root node.

**Save data**
This option is enabled if the page source is an external file (XML, HTML, or DB). It selects whether data is saved on (i) leaving the page, (ii) exiting the entire solution, (iii) when the user clicks the last Submit option, or (iv) when expressly defined as a page or control action. See Page Source Actions.
7.3 Page Source Trees

Tree structure
A page source has an XML tree structure. The screenshot below shows the XML tree structure of a page source that is a DB table.

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

- Imported from an XML or HTML document, or an XQuery statement, when a page source is added (see Page Sources), 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 page source. This file must have a structure that corresponds to the structure of the page source. Its data is then used as the data of the page source.
- A node can be given a fixed value (via the **Ensure exists (fixed value)** command in the node's context menu). This value overrides any value imported from a default file.
- A node is assigned an XPath expression (via the **Ensure exists (XPath value)** command in the node's context menu). The XPath expression generates the content of the node. This value overrides any value imported from a default file.
- A node is updated when an event is defined to trigger an **Update node** action, or when a node is the source node of a control that provides editing functionality (for example, the combo box and edit field controls).

**Page source links**

A **source node (or page source link)** is a page source tree node that is associated with a control. In the cases of controls that provide editing functionality—such as the combo box and edit field controls—data edited
by the end user is passed to the tree node. A source node is assigned to a control by dragging it from the Page Sources Pane onto the control.

The page source link of a control is displayed in bold in the page source tree. When you hover over a page source link, a popup provides information about the associated control/s in the design. Conversely, controls that are associated with a page source link have an icon at the control's top left. Hovering over the icon displays information about the associated page source link.

### 7.3.1 Root Nodes

Each page source is conceptualized as a tree. The root node of each tree is identified by a variable name that is unique within each MobileTogether design (project). The terminology used to address and describe the basic parts of page source trees is given below. A listing of all the page source variables of a project, together with their uses in controls and actions, can be displayed in the Messages window when you click the command 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: $XML/root/element1.

### Names of root nodes

The names of root nodes are generated automatically when page sources are added. The screenshot below, for example, shows two root nodes, named $PERSISTENT and $XML. 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

<table>
<thead>
<tr>
<th>Page Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$PERSISTENT</strong></td>
<td>The $PERSISTENT tree is the tree saved on the client.</td>
</tr>
<tr>
<td>- 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.</td>
<td></td>
</tr>
<tr>
<td>- 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).</td>
<td></td>
</tr>
<tr>
<td>- $PERSISTENT tree nodes that are assigned (as page source links) to a control will be updated on the client. This means that when a solution is loaded in the client, nodes that are assigned to a control will take their data from the $PERSISTENT tree—and not from other page sources.</td>
<td></td>
</tr>
<tr>
<td><strong>$XML</strong></td>
<td>XML documents, created manually or imported. Default data file is optional.</td>
</tr>
<tr>
<td><strong>$HTML</strong></td>
<td>HTML documents, created manually or imported. Default data file is optional.</td>
</tr>
<tr>
<td><strong>$HTTP</strong></td>
<td>Documents accessed via HTTP/FTP, REST, or SOAP. The requested resource provides the data.</td>
</tr>
<tr>
<td><strong>$DB</strong></td>
<td>The selected database table provides both structure and data.</td>
</tr>
<tr>
<td><strong>$XQ</strong></td>
<td>XQuery documents.</td>
</tr>
<tr>
<td><strong>$FLOWFORCE</strong></td>
<td>FlowForce jobs.</td>
</tr>
<tr>
<td><strong>$MT_CONTACTS</strong></td>
<td>Added when a Read Contacts action is added. Filled with data from the client's address book.</td>
</tr>
<tr>
<td><strong>$MT CALENDAR</strong></td>
<td>Added when an Access Calendar action is added. Filled with data from the client's calendars.</td>
</tr>
</tbody>
</table>
### Page Sources (Data Sources)

<table>
<thead>
<tr>
<th>Node Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MT_DBSTRUCTURE</td>
<td>Added when a <a href="#">DB Read Structure</a> action is added. Has a standard structure. Nodes are filled at runtime with data from the DB that is read by the action.</td>
</tr>
<tr>
<td>$MT_EMBEDDEDMESSAGE</td>
<td>Added to the design when the <a href="#">onEmbeddedMessage</a> event is activated, or when an <a href="#">Embedded Message Bck</a> action is added.</td>
</tr>
<tr>
<td>$MT_FILEINFO</td>
<td>Added when a <a href="#">Get File Info</a> or <a href="#">Read Folder</a> action is added. Holds information about the file/s and/or folder/s that are specified in these actions.</td>
</tr>
<tr>
<td>$MT_GEOLOCATION</td>
<td>Added when the <a href="#">Start/Stop Geo Tracking</a> action or the <a href="#">Read Geo Data</a> action is added to design.</td>
</tr>
<tr>
<td>$MT_IN_APP_PURCHASE</td>
<td>Added when the first <a href="#">In-App Purchase action</a> is added to the design. It stores app store data about products and purchases. See <a href="#">In-App-Purchase Page Source</a> for details.</td>
</tr>
<tr>
<td>$MT_NFC</td>
<td>Added when <a href="#">NFC Start/Stop</a> action is added. Filled with data from last discovered NFC tag.</td>
</tr>
<tr>
<td>$MT_PUSHNOTIFICATION</td>
<td>Created at design time when an action is added to the <a href="#">OnPushNotificationReceived</a> event. Filled with data from the payload of a received push notification (PN). See <a href="#">PNs: The Receiving Solution</a>.</td>
</tr>
<tr>
<td>$MT_SERVICE</td>
<td>Created when a <a href="#">service design is created</a>. It holds run time data about service triggers.</td>
</tr>
</tbody>
</table>

**Note:** A root node can be renamed at any time in the design process by double-clicking its name in the Page Sources Pane and editing the name. All references to the old name in XPath expressions throughout the design will be changed to the new name.

### 7.3.2 Namespaces in the Project

Namespaces are important for correctly identifying nodes, and for correctly locating nodes with the use of XPath expressions. The **Namespaces** item in the [Page Sources Pane](#) (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.

### 7.3.3 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 page source, and click the appropriate toolbar command (see toolbar screenshot below). Temporary elements and attributes are intended to hold data used for calculations or data that for any other reason should not be saved to file. The data of temporary nodes is not saved.
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.

### 7.3.4 Tree Data

**Editable and read-only data**

Data in tree nodes can be editable or non-editable (read-only), depending on whether the tree is that of an editable page source or a read-only page source. 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 page source, the end user's combo box selection will be passed to the associated node and become its modified value. In the case of read-only page sources, the content of associated nodes is used for display purposes only. These associated nodes are known as page source links. A source node link is added to a control by dragging the tree node onto the control.

**Assigning data to page sources**

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

- **Assign a default file**: 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 page source can have a default file assigned to it. The data in the default file will be passed to the page source as its data tree. To assign a default file do the following: Just below the root node name of the page source is an entry for the default file (see screenshot below).

![Page Source Tree]

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 page 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 Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the **Working Directory** (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the **Working Directory**. See the section [Location of Project Files](#) for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the **Absolute/Relative Path** field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save**: If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- **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).

  ![Screenshot of the Global Resource Folder Alias with path fragment](image)

  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.

![Specify File dialog](image)

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: **Print To** (Source File and Target File options), **Load/Save File**, **Load/Save Image**, **Load/Save**
Binary File, Load/Save Text File, Read Folder, and Get File Info.

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).

  Note: On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the Sessions setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

Filename is defined by the end user (on the client device)

- **Default file extension for file saving:** When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter:** The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `"txt","html","xml"`).

- **Optional Default File:** You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- **Web Message Box:** Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent
if the action is a file loading action.

**Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

### Adding node content manually

You can manually add content to individual nodes via two commands in the context menu of the selected node:

- **Ensure Exists Before Page Load (Fixed Value):** A fixed string value is added as content of the node and displayed in the tree.
- **Ensure Exists Before Page Load (XPath Value):** An XPath expression provides the content of the node. The XPath expression and Edit XPath icon are displayed in the tree.

The node’s content is generated before the page is loaded, and the page is passed with this node content to the client.

Note that content added manually in this way overrides content added via a default file or with the **Ensure Exists on Load** commands.

### 7.3.5 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 page sources ($XML$, $DB$, $HTML$, etc), some types of page sources have commands that are specific to its type (for example, commands for DB page sources). The specificity of such commands is noted where relevant.

- **Insert, Append, Add Child**

  Enables the addition of elements and attributes relative to the selected node. **Insert** adds the node before the selected node. **Append** adds the node after the last node of that type. If you wish to add a node immediately after the selected node, go to the following node and use the **Insert** command.

- **Delete, Delete from All Pages**

  Deletes the selected page source, respectively, from the current page and from all pages of the project.

- **Keep data on**

  To reduce the amount of data transmitted over the mobile data network—which improves the performance of any mobile solution—MobileTogether lets you specify whether data from the selected page source is transmitted to client devices and/or kept on the server. For example, if a certain data set is only
necessary to display a graph, then that data can be kept on the server. The image of the graph (say in PNG format) will be rendered by the server and transmitted to the client without the underlying data being transferred over the mobile network. For large data sets this produces a significant performance boost.

This command specifies whether data in the tree is stored on the server only, the client only, or shared by both. Note that if data is stored on the server, then it cannot be defined as persistent. See the **Persist data on client** command below.

**Read only data**

A toggle command which specifies that data in the tree is read only. A read-only data tree is used to provide data for display and calculations. It cannot be used to hold data that needs to be edited.

**Persist data on client**

A toggle command that makes a tree a persistent tree. Any number of trees can be defined as persistent. When a tree is defined as persistent, data in it is retained on the client device after the solution is exited. When the solution is opened the next time on that client, the persistent data is displayed. If a tree is defined as persistent, then it cannot be stored on the server. See the **Keep data on server** command above.

**Load data**

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

- **On First Use**: Loads the tree on entering a page where it is used. Once loaded, it will not automatically be reloaded anymore. If you share the same tree on multiple pages, then the first time one of such pages is opened (doesn't matter whether it is a Top page or Sub page), the tree will be loaded and will remain in memory.
- **On Every Page**: Reloads the tree every time you open a page containing the tree, whether the page is a Top page or Sub page. You must be careful with this option: It can slow the processing if loading takes significant time. But this will ensure that the data is retrieved afresh for every page.
- **Not Automatically**: The tree will not automatically be loaded for you. You must use the **Reload**, **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 you to specify whether the data of the selected root node (page source) is automatically reset when a page is left. The following options are available:

- **On Every Page Leave**: The page source is reset on every page leave. Whenever a page is left, the page source will be reset—even if another page that is referencing the page source is still in use.
- **On Last Referencing Page Leave**: The page source is reset when the last page that references it is left. Essentially, a page source is reset only at the point when it is no longer in use by any page. Note that, if a sub page returns to a top page and both pages uses the same page source, then the page source will not be reset.
- **Not Automatically**: The page source is not automatically reset when the page is left. If a data reset is required in a situation not involving a page-leave, then use the **Reset** action.

The default is **Not Automatically**.

**Set as page XPath context**

Sets the selected node as the XPath context node of the page. An annotation to the effect is displayed below the node (see screenshot below). The XPath context of the page is the context node for all XPath expressions on the page.
This command can be toggled on and off. So, you can enable a node as the XPath context node of the page, or you can toggle off a node's setting as the XPath context node of the page. If a node is set as the XPath context node when another node already has this setting, then the setting is toggled off for the previously assigned node and toggled on for the newly assigned node.

Choose default file [root nodes with default files]

Displays the Specify File dialog (screenshot below), in which you specify the file to use as the default file. Data from the default file will be used as the data of the page source. However, in order for the data to be used, the default file must have the same structure as the page source. Note that, when a default file is assigned to a page source, its structure is not automatically imported. To import the structure of the XML file, use the context menu command Import Structure from XML; see below. You can also manually create the structure of the page source 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.
**Specify File**

- **Absolute/Relative Path:** You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server’s database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the **Working Directory** (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the **Working Directory**. See the section Location of Project Files for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the **Absolute/Relative Path** field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- **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).

![Screenshot](image)

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.

![Screenshot](image)

The file to load/save can be specified by you, the designer, or it can be specified by the end user. If you specify the file, then this information will be stored in the solution, and the file will be loaded/saved when the action is triggered. If you choose to let the end user select the file to be loaded/saved, then, when the action is triggered, a browse dialog is opened on the client device and the end user can enter/select the
file to load/save.

**Note:** The option to let the end user select the file to load/save is available for the following actions: Print To (Source File and Target File options), Load/Save File, Load/Save Image, Load/Save Binary File, Load/Save Text File, Read Folder, and Get File Info.

**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).

**Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the Sessions setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

*Filename is defined by the end user (on the client device)*

- **Default file extension for file saving:** When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter:** The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: txt,html;xml), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: ['txt','html','xml']).

- **Optional Default File:** You can enter a default filename, either directly or via an XPath expression, to guide the end user.
• **Web Message Box:** Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

• **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

▼ **View Default File in XMLSpy**

This command is enabled when the root node of an XML page source is selected that has a default file assigned to it. Selecting it opens the default XML file in Altova’s XMLSpy application, which enables you to work directly on the XML file while using the powerful editing and processing features of XMLSpy.

▼ **Embed XML in design file**

This command is enabled when the root node 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 page source) by other pages in the design, then you are prompted to choose from the following options:

- **Modify Shared Structure**: The modifications you are about to make to the page source structure on this page will be shared on the other pages where this DB structure is used.
- **Copy Structure**: The structure is copied to a new page source, and its root element is given a different name. The original page source is removed. The new data structure is not shared any more with any structure on other pages. You can now modify this page source while leaving page sources on other pages unchanged.
- **Cancel**: Cancels the modification process.

Choose DB tables and views  [$DB only]

This command is enabled for database type ($DB) root nodes. It opens MobileTogether Designer's Database Object Selector window, in which you can select the DB tables and views to import as a page source.

Create OriginalRowSet  [$DB only]

In order for data to be edited and saved, the tree of the page source must also include an OriginalRowSet element, which is a copy of the RowSet element. The original data is saved in the OriginalRowSet element, while edited data is saved in the RowSet element. When the page source is saved, the difference between the two trees, OriginalRowSet and RowSet, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to OriginalRowSet so that OriginalRowSet contains the newly saved DB data, and the modification process can be repeated.

Note the following points:

- The OriginalRowSet element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command **Create OriginalRowSet**.
- The **Create OriginalRowSet** command is enabled for database type ($DB) root nodes. It is a toggle command that creates/removes an OriginalRowSet data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the OriginalRowSet structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function `mt-db-original-row`.

Filter columns  [$DB only]
This command is enabled for database type (SDB) 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 page source. You can specify which columns can be updated or can take inserted values. (Updates refer to modified data in existing row elements; inserted values refer to data in newly added row elements.) By default, the Insert and Update options of each column are selected together as a pair. If, however, you wish to specify different options for a column's Insert and Update options, check the Use separate filter settings for insert and update statements check box. Attributes with empty values can be converted to NULL values in the DB by checking that column's NULL check box. Note that missing attributes will always be saved as NULL.

Columns that cannot be updated (because they are user-defined, fixed-value, or calculated-value) will not have an Insert, Update, or NULL option check box. In the screenshot above, the ID column cannot be updated because it holds fixed values. Deselect the columns you do not want to update.

You can specify the order in which deletions, updates, and insertions occur by selecting the desired order in the combo box at the bottom of the dialog.

If you wish to reset the Save settings so that all columns are updated, click Reset to default.

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 (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 page sources). The specificity of such commands is noted where relevant.

**Insert, Append, Add Child**

Enables the addition of elements and attributes relative to the selected node. **Insert** adds the node before the selected node. **Append** adds the node after the last node of that type. If you wish to add a node immediately after the selected node, go to the following node and use the **Insert** command.

**Ensure exists on load (fixed value)**

This context menu item is available for tree nodes. A fixed value can be provided for the selected node when the page is loaded. Click the command and enter the value. This command is a toggle command. So, clicking it when a fixed value is already assigned will remove the value.

**Ensure exists on load (XPath value)**

This context menu item is available for tree nodes. An XPath-generated value can be provided for the selected node when the page is loaded. On clicking the command, the **Edit XPath/XQuery Expression Dialog** 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:

```xquery
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.
7.4 Caches

If a page source gets its data from an XML file or DB, then that page source can be cached on the server. Such a page source has a Cache icon next to its name (see screenshot below). To create a cache for a page source, click the Cache icon, and, in the dialog that appears, configure the cache. A green Cache icon indicates that a cache has been created for the page source; a red icon indicates that no cache exists for the page source (see screenshot below). If a page source is not linked to an XML file or a DB, then it has no cache symbol (as for $XML1 in the screenshot below).

There are two main reasons to create caches:

- If a page source generates reports slowly (for example, a large database)
- If a page source is not modified often. In both these cases, execution of a solution would be faster if data is taken from data caches on the server.

Note the following points:

- If a page source is cached, then the cache will be used when the solution is run. This speeds up solution execution.
- A page source that is based on (i) a DB query that has no parameters, or (ii) an XML file will have one entry in its cache, which can be updated to contain the latest data in the external resource. Update times are specified in the cache configuration.
- A page source that is based on a DB query that has parameters can have multiple cache entries. Each cache entry corresponds to a different parameter combination. At configured update times, all defined cache entries will be updated.
- If a cache exists for a page source, then the cached data will be used when the solution is run.
- Caches can be used as soon as the solution has been deployed to the server.
- If you wish to not use cached data in a solution, you can deactivate the cache in the cache's configuration settings.
- You can also delete a cache in (i) the Cache Overview dialog of MobileTogether Designer, or the Cache Overview tab of MobileTogether Server.
- Once a cache has been defined in MobileTogether Designer, it can be used by the page sources of other designs, provided that the underlying data structure is compatible.
Working with caches involves two broad mechanisms, which are described in the sub-sections of this section

- Creating and configuring caches
- Managing and editing caches

### 7.4.1 Creating Caches

To create a cache for a page source, or to edit the settings of a cache that has already been created, click the **Cache** icon of the page source or select **Cache Settings** from the context menu of the root node of the page source. This displays the Configure Caching dialog *(screenshot below)*. Set the properties of the cache as described below the screenshot, and click **Save** (to save the cache and its properties) or **Save and Fill Cache** (to save the cache as well as fill it with data from the data source).

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

This dialog contains the following settings and controls:
• **Name:** The name of the cache that is being created or edited.

• **Refresh button** (located on the right-hand side of the Name field): Updates the connection to the server to check for the latest caches that match the structure of the current page source. If one or more caches have already been defined for the current page source, then the Name entry field becomes a combo box that enables you to select a cache from the available caches. If you wish to edit the selected cache, click **Change Cache Settings**. If you wish to create a new cache for the current page source, then click the **Add New Cache** button (located on the right-hand side of the Name field).

• **Add New Cache button** (located on the right-hand side of the Name field): Adds a new cache for the current page source and enters a default cache name in the Name field. You can edit the name.

• **Change Cache Settings**: Enabled when an existing cache is selected in the Name field. Click to load the selected cache’s settings and edit these. Click **Save** (at the bottom of the dialog) to save the modified settings.

• **Clear Cache**: Clear the cache on the server.

• **Active**: Activate/deactivate the cache.

• **Show an error**: Specify whether an error should be displayed if the cache is missing or is deactivated when the solution is run.

• **Save data in cache on first request**: Data is saved from the data source when the solution first requests the data.

• **Update cache periodically**: Select to specify the frequency with which the cache should be filled.

• **Save**: Saves the cache and its properties.

• **Save and Fill Cache**: Saves the cache and fills it with data from the data source.

### Maximum cache entries

If the page source that is being cached is a database that has a `SELECT` statement containing one or more parameters, then you can set a maximum number of cache entries. This number specifies how many cache entries will be stored before the first cache entry is deleted and the latest cache entry is appended. This setting: (i) enables a new cache entry to be created each time a new parameter value is found, and (ii) limits the amount of server space that will be used for caches of this particular page source. In order to judge the amount of space that will be used, you can check the size of the cache in the Cache Overview dialog (see below) or in the Cache Overview tab of MobileTogether Server.

### 7.4.2 Cache Overview

The Cache Overview dialog (screenshot below) is accessed with the menu command **Project | Cache Overview**. The Cache tab of MobileTogether Server also provides a cache overview and the same features as the Cache Overview dialog described here.
The dialog provides, in the top pane, an overview of all the caches on the server. In the dialog, you can do the following:

- Activate/deactivate a cache (in the top pane).
- Delete a cache (in the top pane) by selecting it and clicking Delete.
- Select a cache in the top pane to display its details in the other (subsidiary) panes of the dialog.
- Delete a cache item in a subsidiary pane by selecting it and clicking the subsidiary pane's Delete button.
- Reconfigure a cache by clicking the Additional Dialog button (at the right-hand side of the cache entry in the top pane), and editing the cache's configuration.
- View a log of cache entry updates by clicking the Additional Dialog button in the Cache Entries pane (located at the bottom right of the dialog). In the screenshot above the MyCars cache has three cache entries, which are listed in the Cache Entries pane.
8  Pages and Page Events

A MobileTogether project consists of one or more pages, which are defined in an ordered sequence. When the MobileTogether application is started on the client device, the workflow starts at the first page of the sequence and progresses through the sequence to the last page. Top-level pages of the workflow sequence can branch off to subpages and then return to the main workflow. A project containing multiple pages thus enables you to structure a complex workflow into parts that are more comprehensible to the user. The pages of a project are managed in the Pages Pane. 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 as page sources. After the page sources have been defined, they can be used by the controls and actions of that page. For an overview of how these “page sources” are to be used, see the subsection Data Sources of a Page.

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 page 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, Tabbed Pages, and Sub Pages
- Data Sources of a Page
- Page Properties
- Page Events
8.1 Pages, Tabbed Pages, and Sub Pages

A MobileTogether project consists of one or more pages, which are defined in an ordered sequence. When the MobileTogether application is started on the client device, the workflow starts at the first page of the sequence and progresses through the sequence to the last page. Top-level pages of the workflow sequence can branch off to subpages and then return to the main workflow. A project containing multiple pages thus enables you to structure a complex workflow into parts that are more comprehensible to the user.

Pages Pane: managing the pages of a project

The pages of a project are managed in the Pages Pane (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.

You can declare parameters for a sub page, as well as define variables that will be in scope in that sub page. The difference between parameters and variables is as follows:

- The values of parameters are defined externally and are evaluated when the sub page is called. In the Go to Subpage action, for example, when you select a sub page, the parameters of that sub page are displayed and you can define a value for each parameter. The sub page will then be passed the parameter values defined in the action.
- The values of variables, on the other hand, are defined on the sub page itself. This XPath expression can use the parameters of that control template as well as variables defined earlier in the list of variable definitions. For example: If a sub page has parameters $a, $b, $c, and variables $x, $y, $z (in that order), then the variable $y can use the following list of parameters and variables to generate its value: $a, $b, $c, $x, (but not $z).

Note: You can specify whether a parameter or variable is mandatory or optional and that it is an error if no value is passed to a mandatory parameter. To edit define and edit the definitions of parameters and variables, click the Add Parameters/Variables button (which is located to the right of the sub page name) and define the parameters/variables in the Parameters dialog that appears.

Note: You can change the order of parameters and variables by dragging a parameter/variable and dropping it to a new location.

Note: If you change the order of parameters, you need to make sure that the parameter order is identical in the call to the subpage. Do this as follows: (i) Right-click the sub page and select List Usages; (ii) In each usage (the list is shown in the Listings Pane), check whether parameters are specified via an array or a list; (iii) If specified via a list, make sure that the order of parameters in the array is the same as that in the parameter definitions list (i.e., the Parameters dialog).

For more information about managing pages, see the description of the Pages Pane.
8.2 Data Sources of a Page

Every page of a design has a set of page sources, which are managed in the Page Sources Pane. To add a page source, click the pane's Add Source toolbar icon. This displays the Add Page Source dialog (screenshot below). Here you select the type of the page source you want to add. If you want to add the same page source as one that you have used in an already existing page, then select the Reuse existing structure option and select from the existing page sources that are displayed in the combo box.

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

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

For more detailed information about page sources, see the sections Page Sources Pane and Page Sources (Data Sources).
8.3 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.

- **All Styles**
  The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

  The property enables you to set all page styles via a single XPath map expression, such as the two map expressions below:

  ```xml
  map{
    "BackgroundColor" : $XML1/R/@background,
    "Margin"        : "6dp"
  }

  map{
    "Style Sheet" : "Sheet-1"
  }
  ```

  Note the following points:
• When you enter a value for the **All Styles** property—even if it is not a map—all the styling properties of the page in the [Styles & Properties Pane](#) will no longer be visible.
• This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
• You can enter the key–value pairs of the map in any order.
• The key names are the names of style properties (or styles). In the first map above, for example, **Background Color** and **Margin** are style names. The styles that are available for a particular component are listed under that component in the [Styles & Properties Pane](#).
• You can also specify a style sheet to use, as shown in the second map above.

### Show Page Title Bar
Specifies whether the title bar of the page is shown (true) or not (false). The default value is true. Note that this property refers to the presence or absence of the entire title bar. It does not refer to the content of the title bar, which is specified by the **Page Title** property.

### Page Title
The title of the page in the solution. Click inside the value field and enter the name you want. Alternatively, you can enter an XPath expression by clicking the **XPath** icon in the pane's toolbar. If a value for this property does not exist, then the value of the **Name** property will be used as the title of the page in the solution.

### Auto-Add Submit Button
A boolean setting that defines whether a **Submit** button is automatically added to the page. Select true or false in the combo box. The default value is true. (The **Submit** button of a page in the solution is usually located at the top right of the page, and it submits data on the page for action. Typically, the workflow then moves on to the next page.)

### Submit on Assertion
Allows or disallows a page submission if there are invalid assertions on the page. Select from the following values:

- **Disable**: The **Submit** button is disabled if there is an invalid assertion on the page. This is the default setting.
- **Enable**: The **Submit** button is enabled even if there is an invalid assertion on the page.
- **Ask**: The **Submit** button is enabled even if there is an invalid assertion on the page. However, if there is an invalid assertion, and the **Submit** button is clicked/tapped, then a dialog appears asking the end-user whether submission should proceed or not.

The default is **Disable**.

### Page Actions
Clicking the property's **Additional Dialog** button displays the [Page Actions dialog](#), 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

### Margin
Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

- Points versus pixels on iOS devices
  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Style Sheet
The Style Sheet property sets the style sheet to use for the project. 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 Max Width
Specifies the maximum page width in browsers. Select a value from the dropdown list, or click the window's XPath toolbar icon to specify the width via an XPath expression. Values can be given as (i) a
percentage of the browser width, (ii) a dp value (density-independent pixels), (iii) an sp value (scale-independent pixels), or (iv) an absolute pixel value. The default value is the available browser width. Note that the page width will be limited to the maximum you specify only if the screen width is larger than the specified value. See Sizes: Pixels, DPI, DP, SP for more information.

**Browser CSS Class**

Enter the name of the CSS class that you want to associate with this page. This class can then be used in a CSS file (specified in the Project Properties) to assign properties for this control separately. A value of `mt-no-browser-exit-confirmation` for this property (see screenshot above) disables the page-exit confirmation dialog from appearing for this page—if the page-exit confirmation option has been set in the Browser Settings of the project (that is, for all pages).
8.4 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.

8.4.1 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.
8.4.2 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.
- Whenever the orientation (portrait or landscape) is changed, or the app window is re-sized.

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 page source node, or a value that is generated by a calculation).
• The timer is first started when the page is loaded. The On Timer Refresh actions are executed at intervals determined from this start time onwards. If you change the refresh interval, then the timer must be restarted. This is done by adding the Restart Page Timer 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.

**Refresh due to orientation change or resizing of app window**

The actions defined for this option are executed when the end user changes device orientation (between portrait and landscape) or re-sizes the app window (on devices where the window is re-sizable). For example, if the device orientation is changed from landscape to portrait, you can reduce the number of columns in a table. The following variables are particularly useful for the actions defined here: device-dimension variables, device-orientation variables, variables for device viewport dimensions and variables for window size.

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 \( \times \) seconds, where \( \times \) 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.
8.4.3 **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.
8.4.4 OnSubmitButtonClick

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 OnSubmitButtonClick event of the page.

8.4.5 OnServerErrorConnection

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 OnServerErrorConnection 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 OnServerErrorConnection. 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)*.

   ![Server Connection Error]

   Server Connection Error: Try again after a few minutes. If the problem persists, contact your MobileTogether administrator.

   ![OK]

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).
1. Click **Prevent Server Access**. Server access will be disabled, and the button will toggle into an **Enable Server Access** button.

2. Carry out an action that calls for a server connection. Since access is disabled, the actions defined in the **OnServerConnectionError** page event are triggered.

3. To enable server access again, click **Enable Server Access** in the simulator.

### 8.4.6 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.
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.
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.

![Screenshot of MobileTogether Designer](image-url)

**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 page source node** (available in the Page Sources Pane) so that data from a node in the page source can be displayed in the design and processed. Typically, an association between a control and node is defined by dragging a data node from its source tree (in the Page Sources Pane) 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 its 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

© 2015-2021 Altova GmbH
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\(^2\). After a control has been placed in the design, you can do the following:

- **Controls can be linked to page source nodes** (available in the Page Sources Pane\(^3\)) so that data from these nodes can be displayed in the design and processed. Typically, an association between a control and a page source node is defined by dragging a node from its source tree (in the Page Sources Pane\(^3\)) onto the control. A node association of this type is called the page source link of the control.

- **Set control actions**: Control actions\(^4\) determine the functionality to be executed when a control-related event is triggered (for example, when a button is clicked). You define control actions\(^4\) by right-clicking a control, and selecting Control Actions. Control actions\(^4\) are described in the section Actions\(^4\).

- **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\(^5\). 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\(^6\)) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View\(^6\)) and click Delete Page Source Link.\(^6\)
- To reset a style or property (in the Styles & Properties Pane\(^9\)), select the property and click Reset in the pane's toolbar.\(^9\)
- 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.\(^9\)
- To edit the XPath expression of a style or property (in the Styles & Properties Pane\(^9\)), select the style or property, and click Edit XPath in the pane's toolbar.\(^9\)
- 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\(^9\)).
- A control's CSS properties can be defined in the Styles & Properties Pane\(^9\) and/or in an external CSS file. Those defined in the Styles & Properties Pane\(^9\) have priority.

**List of controls**

Given below is a list of available controls, arranged alphabetically, together with a screenshot showing different controls..
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
Controls and Control Events

- **Description**
  This command is enabled for those controls that can be associated with a page source node and for which an association exists. **Delete Page Source Link** deletes the association between control and page source. Note that there is no other way to delete the control's association with a node.

- **Control Actions**
  - **Description**
    Displays the **Control Actions dialog**, 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 page 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 \l 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 \l Page Actions command.
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.

9.1.1 Assertion Message

The Assertion Message control displays the assertion message of the first invalid assertion of the page. An Assertion is a property of a page and of some—not all—controls. It specifies a certain condition (for example that a node may not be empty). If the Assertion property's condition is not met, the assertion is invalid, and the Assertion Message property associated with that Assertion property is displayed in the Assertion Message control.

An Assertion Message control can be placed anywhere in the design. It will always display the Assertion Message property text that is associated with the first invalid assertion on the page. If there are multiple invalid assertions on a page, then the assertion message of the first invalid assertion (in the sequence in which assertions appear on the page) is displayed. The Assertion Message control should therefore be inserted only once on a page. If multiple Assertion Message controls are placed in the design, they will all display the same assertion message (that of the first invalid assertion).

Assertions and assertion messages work as follows:

- The Assertion property of a control or page sets a condition to be met in order for the assertion to be valid. The assertion's condition is specified with an XPath expression.
- If the assertion is invalid, then the text of the control's Assertion Message property is displayed in the Assertion Message 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.

▼ All Styles

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
  "Bold Text"    : $XML1/R/@bold = "1",
  "Italic Text" : true(),
  "Text"         : "hello",
  "Text Color"   : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size"   : $XML1/R/@textsize
}
map{
  "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Multiline

Sets multiline input/display on or off (true/false). The default is false. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

**Note**

- If this property is set to true, and if the Text Size Auto-Fit property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the Max Number of Lines property becomes available.
- Check boxes with multilines can be vertically aligned via the Vertical Alignment property.
**Text Color**

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a text color for when the control is disabled, use the **Text Color (Disabled)** property.

**Text Size**

Select a size from the dropdown list of the combo box. Allowed values are: smallest | small | medium | large | largest. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value medium.

You can generate other values by using the `mt-font-height` function. For example, to get a size that is 120% larger than the numeric size that corresponds to `largest` on a device, use the following XPath expression for the `TextSize` value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the `largest` size. This value is then multiplied by 1.2 to obtain the numeric value that is 120% of the value that corresponds to `largest`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Text Size Auto-Fit**
Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- **off** *(the text is not automatically resized; this is the default)*
- **ellipsis** *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- **individually** *(the text of this control only will be automatically resized)*
- **group X** *(where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.*
- **template group X** *(where X=1 to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.*

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**
- If the Multiline property has been set to true, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table’s Wrap Content Auto-Fit Group property.

![Bold Text](image)

Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

![Italic Text](image)

Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

![Underline Text](image)

Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

![Background Color](image)

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; to select the background color of the
control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

**Tooltip**

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Control Width**

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent:** makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content:** makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- **wrap_content_longest_entry:** is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- **percent value:** a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value:** select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, fill_parent creates a maximum width, while wrap_content creates a minimum width. If the combo box is within a table cell, for example, fill_parent would let the combo box fill the cell whereas wrap_content might not fill the cell.
The default value is *fill_parent* for all controls except the following:

- **Image and Chart**: For these, the default is *wrap_content*.
- **Geolocation Map**: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device’s viewport. Since the default of both *Control Height* and *Control Width* are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to $\min(MT_CanvasX, MT_CanvasY)$.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as *points* in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX$ and $MT_CanvasY$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be `concat(MT_CanvasX * 0.5, 'px')`.

**Max Control Width**

This property is available only when the control’s *Control Width* property has been set to *wrap_content*. The **Max Control Width** property sets the maximum width of the control. Select a value from the property’s combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

**Margin**

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the **Margin** property to display the individual margin properties (left, right, top and bottom), and set the different value. For
example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasWidth and $MT_CanvasHeight dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasWidth * 0.5; the XPath expression for this image width would be concat($MT_CanvasWidth * 0.5, 'px').

Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.
In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: the pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Style Sheet**

The Style Sheet property sets the style sheet to use for the control. 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).

### 9.1.2 Button

Buttons can be used to execute an action when the button is clicked. The name of the button can be static text (entered as the value of the Text property; see below) or a dynamic value obtained from a page source node (by dragging the node onto the button). You can also alternatively or additionally select an icon from the options available in the Button Image property combo box. The height of the button is determined by (i) the height of the button icon or text, whichever is greater, and (ii) the vertical padding set on the button. The OnButtonClicked event is associated with the button control. To define an action for this event, click the Additional Dialog button of the Control Action property. This displays the Control Actions dialog, 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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) and click Delete Page Source Link.
- To reset a style or property (in the Styles & Properties Pane), select the property and click Reset.
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 (**On Click**) or a longer press (**On Long Click**). A sequence of different actions can be specified for each type of click (see screenshots at left and middle below). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions be executed after those of the end-user click; these actions are defined after the **On Long Click** event (see screenshot below right).
  - **On Click**: The action/s to perform when the control is tapped (see screenshot above left).
  - **On Long Click**: The action/s to perform when the control is pressed for a longer time than a tap (see screenshot above center).
  - **Additional actions**: The action/s to perform after the **On Click** or **On Long Click** actions have been executed (see screenshot above right). If no action has been set for the **On Click** or **On Long Click** events, then the additional action/s are performed directly on a click or long-click.

You can combine actions for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.
The screenshot above shows that the On Click and On Long Click events each has a sequence of actions defined for it. An additional MessageBox event is defined after the On Long Click event. This MessageBox event will be executed after the On Click or On Long Click sequence of actions has completed.

**On Enter/Escape**

If the control's On Enter or On Escape check box is selected, then the control's actions are executed when the respective key (Enter or Escape) is tapped. The key-tap (Enter or Escape) serves as an alternative to the On Click event, and will work additionally to the click. The screenshot below shows the On Enter and On Escape check boxes of the Button event. Other controls that provide this option look similar and work similarly.

This setting can also be accessed via the control's On Enter/Escape property, which is described below.

**Note:** If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an Enter or Escape key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

### Button properties

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

- **Name**

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

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
}

map{
    "Style Sheet" : "Sheet-1"
}

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

Text

The Text property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the XPath toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

Note: In the Rich Text control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text section for more information.

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

▼ Multiline
Sets multiline input/display on or off (true/false). The default is false. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

**Note**
- If this property is set to true, and if the Text Size Auto-Fit property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the Max Number of Lines property becomes available.
- Check boxes with multilines can be vertically aligned via the Vertical Alignment property.

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

![Format dialog screenshot](image_url)

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

**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. If the names of months and weekdays are used in the format string, then these will be displayed in the language selected for this property. The default language is English.

**Button Image**

Adds a predefined icon or a custom image to the button display. You can also optionally specify (in the Text property) an additional text string to accompany the button icon. If you use both icon and text, then the Button Image Position property specifies whether the icon is positioned to the left or the right of the text, and the Button Image/Text Distance property specifies the distance in pixels between the button and the text. Horizontal positioning of the icon-text pair within the button is specified with the Horizontal Alignment property.

Add a predefined icon by selecting one from the dropdown list of the property's combo box. Available icons include: +, –, >, Calendar, Closed Tree, Copy, Cut, Delete, Edit, Email, Export, Fast Forward, Fast Rewind, Import, Microphone, Offline, Opened Tree, Paste, Pause, Play, Play Reverse, Print, Print PDF, Print Word, Redo, Refresh, Report, Resume, Search, Settings, Share, Snooze, Stop, Stop Playing, Time,
**Undo and View.**

Add a custom image by selecting *Custom Image*. On doing this, two related properties become available: Image Source and Image Source Type *(see their descriptions below).*

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is any one of the values listed above. *(For users of localized (non-English) MobileTogether Designer editions: Note that the expression must evaluate to the English-language value, not the localized value.)*

The default value of the Button Image property is no value, which results in no icon being displayed.

When a predefined icon or *Custom Image* is selected, then the Button Image Color, Button Image Color (Disabled), Button Background, Button Image Position, and Button Image/Text Distance properties become available. These enable you to set, respectively, (i) a transparent or non-transparent background for the button, and (ii) the horizontal position of the icon relative to button text.

**Button Image Color**

The Button Image Color property is enabled when the Button Image property has been set to a predefined image or custom image. It sets the color of the button image when the button is enabled. The default value is client-specific.

**Button Image Color (Disabled)**

The Button Image Color (Disabled) property is enabled when the Button Image property has been set to a predefined button image or custom image. It sets the color of the button image when the button is disabled. The condition/s for enabling and disabling a button is set on the button's Enabled/Editable property. The property's default value is client-specific.

**Button Background**

Selects whether the background of the button icon is transparent or non-transparent. The default is not transparent.

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is either transparent or not transparent. *(For users of localized (non-English) MobileTogether Designer editions: Note that the expression must evaluate to the English-language value, not the localized value.)*

**Button Image Position**

Specifies whether the button image (predefined icon or custom image) is located to the left or right of the button text (which is specified via the Text property). The default value is left of text.

The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is either left of text or right of text. *(For users of localized (non-English) MobileTogether Designer editions: Note that the expression must evaluate to the English-language value, not the localized value.)*

**Button Image/Text Distance**

Sets the distance, in pixels, dp, or sp between the button image and button text. The default value is the default button padding (which is 0px).
The value of this property can also be entered as an XPath expression that is a string value. The allowed string values are those listed in the property's combo box.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Image Source

The value of the Image Source property references an image in one of the following ways:

- The URL of a binary image file (PNG, BMP, etc). The value of the property must be a URL. The URL is selected in the Specify File dialog (see description below).
- An image file represented as a Base64-encoded string. The value of the property must be a Base64-encoded string. An XPath expression supplies the string, which could be entered directly or obtained from 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 Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the Working Directory (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the Working Directory. See the section Location of Project Files for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the Absolute/Relative Path field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are
missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- **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).

![Screenshot of file alias selection](image)

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 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, Load/Save Binary File, Load/Save Text File, Read Folder, and Get File Info.

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).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the *Sessions* setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

  *Filename is defined by the end user (on the client device)*

- **Default file extension for file saving:** When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter:** The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt, html, xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt'`, `'html'`, `'xml'`).

- **Optional Default File:** You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- **Web Message Box:** Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

▼ **Image Source Type**

Sets the type of the image 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`. 
The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is either url or base64.

**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 event/s are predefined and each is shown in its own 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. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

**Visible**

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

**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. 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'.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5;
the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

▼ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- off (the text is not automatically resized; this is the default)
- ellipsis (adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)
- individually (the text of this control only will be automatically resized)
- group X (where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- template group X (where X=1 to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**
- If the Multiline property has been set to true, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines.
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table’s Wrap Content Auto-Fit Group property.

▼ Bold Text

Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

▼ Italic Text

Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

▼ Underline Text

Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

▼ Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to
select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item *Control's background color* in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's *Enabled/Editable* property. To set a background color for when the object is disabled, use the *Background Color (Disabled)* property.

**Background Color (Disabled)**

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's *Enabled/Editable* property. To set a background color for when the object is enabled, use the *Background Color* property.

**Tooltip**

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).
Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page.
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available.
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available.
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly.
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly.

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the combo box is within a table cell, for example, **fill_parent** would let the combo box fill the cell whereas **wrap_content** might not fill the cell.

The default value is **fill_parent** for all controls except the following:

- **Image** and **Chart**: For these, the default is **wrap_content**.
- **Geolocation Map**: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to \( \min(MT_CanvasX, MT_CanvasY) \).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](sizes_pixels_dp_sp).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX$ and $MT_CanvasY$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be `concat(MT_CanvasX * 0.5, 'px')`.  

Max Control Width

This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the \$MT_CanvasX and \$MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to \$MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value
field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the **Padding** property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set **Padding** to be 6px and **Padding Bottom** to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to **$MT_CanvasX * 0.5**; the XPath expression for this image width would be **concat($MT_CanvasX * 0.5, 'px')**.

**Tab Order**

The **Tab Order** property takes an integer as its value, or an **XPath** expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** menu command. The **Tab Order** property of individual controls sets the sequential position of that control only.

**Note:** The Tab Order feature is available on Web and Windows clients only.

**On Enter/Escape**

Takes one of three values:

- **OnEnter**: Specifies that the actions of this control are executed when the **Enter** key is tapped.
- **OnEscape**: Specifies that the actions of this control are executed when the **Escape** key is tapped.
- None: No action when either **Enter** or **Escape** is pressed. This is the default value.

If **XPath** expressions are used to generate the values, the expressions must evaluate to "**OnEnter**" or "**OnEscape**". If more than one control on a page is given the same value (OnEnter or OnEscape), then the first visible and enabled control that has the value is selected when the key is tapped. (See the **Visible**
and Enabled/Editable properties.)

This setting can also be made via the dialog to set the control's OnClicked actions (see the description of the control's events above).

**Note:** If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an Enter or Escape key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

**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).

### 9.1.3 Chart

The Chart control enables data from a source data file to be displayed in the form of a chart. The available chart types 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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
• To remove a data-source node association (and therefore the data in the control), right-click the control (in **Page Design View**\(^{204}\)) and click **Delete Page Source Link**.

• To reset a style or property (in the **Styles & Properties Pane**\(^{219}\)), select the property and click **Reset** in the pane's toolbar\(^{224}\).

• 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**\(^{228}\).

• To edit the XPath expression of a style or property (in the **Styles & Properties Pane**\(^{219}\)), select the style or property, and click **Edit XPath** in the pane's toolbar\(^{224}\).

• **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**\(^{233}\)).

• A control's CSS properties can be defined in the **Styles & Properties Pane**\(^{219}\) and/or in an external CSS file\(^{232}\). Those defined in the **Styles & Properties Pane**\(^{219}\) have priority.

**Chart events**

The **OnImageClicked** event is available (the image that is clicked is the chart). To define actions for the chart's **OnImageClicked** event, right-click the chart and, from the context menu that appears, select **Control Actions for 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**\(^{238}\).

**OnChartClicked (OnClick, OnLongClick)**

The end user can click the control in one of two ways: a short tap (**On Click**) or a longer press (**On Long Click**). A sequence of different **actions**\(^{239}\) can be specified for each type of click (see **screenshots at left and middle below**). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional **actions**\(^{239}\) be executed after those of the end-user click; these actions are defined after the **On Long Click** event (see **screenshot below right**).\(^{238}\)

- **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**\(^{239}\) for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.
The screenshot above shows that the On Click and On Long Click events each has a sequence of actions defined for it. An additional MessageBox event is defined after the On Long Click event. This MessageBox event will be executed after the On Click or On Long Click sequence of actions has completed.

On Enter/Escape
If the control’s On Enter or On Escape check box is selected, then the control’s actions are executed when the respective key (Enter or Escape) is tapped. The key-tap (Enter or Escape) serves as an alternative to the On Click event, and will work additionally to the click. The screenshot below shows the On Enter and On Escape check boxes of the Button event. Other controls that provide this option look similar and work similarly.

This setting can also be accessed via the control’s On Enter/Escape property, which is described below.

Note: If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an Enter or Escape key-tap are marked with a symbol of the respective key.

Note: This feature is available on Web clients and Windows clients, and in simulations of all clients.

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

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

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map {
    "Bold Text"          : $XML1/R/@bold = "1",
    "Italic Text"        : true(),
    "Text"               : "hello",
    "Text Color"         : "red",
    "Background Color"   : $XML1/R/@background,
    "Text Size"          : $XML1/R/@textsize
}

map {
    "Style Sheet"        : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

Chart Settings

Click the Additional Dialog button to display the Chart Configuration dialog. The settings you make in this dialog will apply to the chart that is currently selected in the design. For a description of how to configure charts, see the section, Charts.

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.
合法权益。 • Charter Creation Width

Sets the width, in pixels, of the chart to be generated. Click the Edit XPath icon and, in the dialog that appears, enter an expression that returns a numeric value. This value will be the width, in pixels, dp or sp, of the chart to be generated.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Chart Creation Height

Sets the height, in pixels, of the chart to be generated. Click the Edit XPath icon and, in the dialog that appears, enter an expression that returns a numeric value. This value will be the height, in pixels, dp or sp, of the chart to be generated.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').
conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $\text{MT}_\text{CanvasX} * 0.5$; the XPath expression for this image width would be `concat($\text{MT}_\text{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 event/s are predefined and each is shown in its own 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. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

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

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Control Width
Controls and Control Events

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the combo box is within a table cell, for example, **fill_parent** would let the combo box fill the cell whereas **wrap_content** might not fill the cell.

The default value is **fill_parent** for all controls except the following:

- **Image and Chart**: For these, the default is **wrap_content**.
- **Geolocation Map**: The default is the smaller of the two values **$MT_CanvasX** and **$MT_CanvasY**. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of **$MT_CanvasX** and **$MT_CanvasY**), the default shape and size of the viewport in the control will always be a square with side equal to **min($MT_CanvasX, $MT_CanvasY)**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see **Sizes: Pixels, DPI, DP, SP**.

- **Points versus pixels on iOS devices**

  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

  In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to **$MT_CanvasX * 0.5**; the XPath expression for this image width would be **concat($MT_CanvasX * 0.5, 'px')**.
Max Control Width

This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

Limit Control Height to Canvas

Select one of the allowed values (true or false) in the combo box. In case the control's height exceeds the device height, a value of true restricts the height to that of the device display. Default is true.

Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top, and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

On Enter/Escape

Takes one of three values:

- **OnEnter**: Specifies that the actions of this control are executed when the Enter key is tapped.
Controls and Control Events

- **OnEscape**: Specifies that the actions of this control are executed when the `Escape` key is tapped.
- **None**: No action when either `Enter` or `Escape` is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to "OnEnter" or "OnEscape". If more than one control on a page is given the same value (OnEnter or OnEscape), then the first visible and enabled control that has the value is selected when the key is tapped. (See the visible and Enabled/Editable properties.)

This setting can also be made via the dialog to set the control's `OnClicked` actions (see the description of the control's events above).

**Note**: If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an `Enter` or `Escape` key-tap are marked with a symbol of the respective key.

**Note**: This feature is available on Web clients and Windows clients, and in simulations of all clients.

### 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).

9.1.4 **Check Box**

Check boxes allow one of two possible values to be entered as the content of a node. In this way the user can be constrained to select one of two specific values. When you insert a check box control, you can specify whether the check box should be to the left or right of the check box text, or in the default system position. Two key properties of the check box control are:

- The text that accompanies the check box. This can be static text (entered as the value of the Text property; see below) or a dynamic value obtained via an XPath expression.
- The values that are to be respectively assigned the checked and unchecked states of the check box. These are assigned with the Checked Values property (see below). To specify which node will receive the value, make a page source link from the check box to a page source node (by dragging the node on to the check box).

Check boxes have the OnFinishEditing event, which is triggered when the end-user makes a check box selection. To define an action for this event, click the Additional Dialog button of the Control Action property. This displays the Control Actions dialog, 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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

- All Styles
  The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)
  The property enables you to set all of the component’s styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
  "Bold Text" : $XML1/R/@bold = "1",
  "Italic Text" : true(),
}
```
"Text" : "hello",
"Text Color" : "red",
"Background Color" : $XML1/R/@background,
"Text Size" : $XML1/R/@textsize
}

map{
  "Style Sheet" : "Sheet-1"
}

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key-value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

**Text**

The **Text** property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the **Rich Text** control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the **Rich Text** section for more information.

**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. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

**Note**

- If this property is set to true, and if the **Text Size Auto-Fit** property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the **Max Number of Lines** property becomes...
Controls and Control Events

- Check boxes with multilines can be vertically aligned via the Vertical Alignment property.

### Checked/Unchecked Values

Provides an XML data value for the selected/unselected state of the control. The defaults are, respectively, 1 and 0. To change the XML values that will be entered in the associated page source node, click the property's Additional Dialog button. In the Edit Checked Values dialog that appears (screenshot below), enter a value for each the checked (selected) and unchecked (unselected) state. When the end user changes the control's state, the value corresponding to the new state (selected/unselected) will be entered in the page source node.

![Edit Checked Values dialog](image)

Note the following points:

- To specify which node will receive the value, make a page source link from the control to a page source node (by dragging the node on to the control).
- If you enter more than one value for the Checked value, then the first value will be used; the others are ignored.
- If you enter a Checked value but do not enter an Unchecked value, then the Unchecked value is no longer the default value but an empty string.
- You can enter values for the Checked/Unchecked Values (true/false) property in a style sheet for: (i) all check boxes, and/or (ii) all switches, and/or (iii) all controls that use Checked/Unchecked Values (true/false) (which are check boxes and switches).
- If values for Checked/Unchecked Values (true/false) are defined in a style sheet, and a control is, as a consequence, assigned values for this property in more than one place, then the most local definition wins. The most local definition is the style defined directly in the control's property. For priority involving a style sheet, see Priority within a Style Sheet and Priority across Style Sheets.

### 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 an XML value. These XML values are defined in the Checked Values property. The control is also associated with a page source node. If the value in the page source node is anything other than the XML value defined for the current status of the control (checked or unchecked), then auto-correction updates the value in the node to the value that has been defined for the current status.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node action. The Auto Correct Value property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

For example, if the checked value is defined to be 1 and the unchecked value to be 0, and if the node's content is "somestring", then, (i) if the control is checked, the node's content will be corrected to 1, (ii) if the control is unchecked, the node's content will be corrected to 0. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction.

The Auto Correct Value property has two possible values: true or false. If the property is set to true, XML values are automatically corrected. The property's default value is false.

▼ Check Mark Position

Sets the position of the check box relative to the control's text: either to the left or right of the text. The default is the operating system default.

▼ Control Action

Click the Additional Dialog button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's Additional Dialog button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

▼ Visible

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false() — and only if it evaluates to false() — then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.
Note: For information about the visibility of spanned columns/rows, see Table Properties.

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. 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’.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows:
The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- **off** *(the text is not automatically resized; this is the default)*
- **ellipsis** *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- **individually** *(the text of this control only will be automatically resized)*
- **group X** *(where X=1 to 9)*. You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- **template group X** *(where X=1 to 9)*. Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit
size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**
- If the Multiline property has been set to true, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table’s Wrap Content Auto-Fit Group property.

**Bold Text**
Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

**Italic Text**
Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

**Underline Text**
Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

**Background Color**
Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

**Background Color (Disabled)**
Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
• Select a color from the dropdown list of the combo box
• Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is enabled, use the **Background Color** property.

**Tooltip**
Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**
This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to **left**, **center**, or **right**. Default is left for all controls except vertical lines, for which it is center. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**
Sets the vertical alignment to **top**, **middle**, or **bottom**. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the **Multiline** property).

**Control Width**
Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the
combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- **Image and Chart**: For these, the default is `wrap_content`.
- **Geolocation Map**: The default is the smaller of the two values `$MT_CanvasX` and `$MT_CanvasY`. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of `$MT_CanvasX` and `$MT_CanvasY`), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see `Sizes: Pixels, DPI, DP, SP`.

### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Max Control Width

This property is available only when the control's `Control Width` property has been set to `wrap_content`. The `Max Control Width` property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

### Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the
control or page. If you wish to set a different margin for any of the four sides, expand the Margin property
to display the individual margin properties (left, right, top and bottom), and set the different value. For
example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins
will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-
independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these
values are read as points in the viewport coordinate space. The viewport coordinate space is the
canvas on which the design components are drawn, and a point is the length unit in this space; a
point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically
maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping
the values in this way (from viewport values to device values) ensures that design components
maintain the same size relationship to both the canvas and to each other, no matter what the
resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic
variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to
these dimensions. (For iOS devices, the values returned by these variables are calculated as follows:
The pixel dimensions of the current device coordinate space are converted (using an appropriate
conversion factor) to point values in the viewport coordinate space. These point values—as numbers
—are returned by the variables as pixels for use in the design.) For example, if you want an image to
be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5;
the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the
control is a table and the padding is set on a table row or column, then the padding is added to the margin
of any control that is contained in the row or column; this increases the inset of the contained control and
does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value
field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a
different padding for any of the four sides, expand the Padding property to display the individual padding
properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be
6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom
padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-
independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these
values are read as points in the viewport coordinate space. The viewport coordinate space is the
canvas on which the design components are drawn, and a point is the length unit in this space; a
point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically
maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping
the values in this way (from viewport values to device values) ensures that design components
maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Tab Order

The **Tab Order** property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the **Tab** key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** 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).

## 9.1.5 Combo Box

The Combo Box control provides a mechanism consisting of two parts:

- **Entry text and XML values:** The text of combo box entries (the dropdown list that the end user sees), and their corresponding XML values (that go into the page source node) are specified with the **Combo Box Entry Values** property (see below for details).
- **Associated XML node:** A node from the **Page Sources Pane** 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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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 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.

#### All Styles

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as
the two map expressions below:

```xml
map{
    "Bold Text"        : $XML1/R/@bold = "1",
    "Italic Text"      : true(),
    "Text"             : "hello",
    "Text Color"       : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size"        : $XML1/R/@textsize
}
map{
    "Style Sheet"      : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

♥ Multiline

Sets multiline input/display on or off (true/false). The default is false. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

**Note**

- If this property is set to true, and if the Text Size Auto-Fit property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the Max Number of Lines property becomes available.
- Check boxes with multilines can be vertically aligned via the Vertical Alignment property.

♥ Combo Box Entry Values

Provides XML data values for the items of the dropdown list of the combo box. Click the Additional Dialog button to display the Edit Combo Box dialog (screenshot below). Alternatively, you can double-click the combo box to display the Edit Combo Box dialog.
To define the entries and values for the combo box, do the following:

1. Select the method with which you wish to define the entries and values by clicking the appropriate radio button to select values: (i) Use list of values, or (ii) Use XPath expression. How to use these methods is described below.
2. If you wish to have the items that appear in the drop-down list of the combo box sorted, check the Sort Values check box.
3. Click OK to finish.

Using a list of values
If you select Use List of Values, you can specify each item of the dropdown list of the combo box as a
text string. You can choose to use the text string for both (i) the text of the item in the dropdown list, as well as (ii) the XML data value of the item. Alternatively, you can specify different text strings for the dropdown list item and for its corresponding XML data value. For each item, add a new row and enter the desired text string/s.

**Using XPath expressions**
If you select *Use XPath expression*, you can use XPath expressions to specify the dropdown list items and the items' corresponding XML data values. There are two alternative ways of doing this:

- Use one XPath expression to define both the visible text (of the item in the dropdown list of the combo box) and the corresponding XML data values. The XPath expression must evaluate to either (i) a sequence of strings, where each string provides the value of both the visible combo box entry as well as that entry's XML data value (example: `("One", "Two", "Three")`), or (ii) a sequence of two-member arrays, where the first member is the XML data value and the second member is the visible entry (example: `[
"One", "1"]`, `[
"Two", "2"]`, `[
"Three", "3"]`).

- Use two different XPath expressions to generate, respectively, (i) the sequence of strings that are the XML data values, and (ii) the sequence of strings that are the visible entries of the combo box. A one-to-one index mapping between the items of the two sequences determines the correspondence of visible entry to XML value. So, for example, the first items of both sequences are mapped to each other, the second items to each other, and so on.

**Note:** Using XPath expressions enables you to generate entries and XML data values dynamically. For example, the expression `distinct-values(region), "All"` generates a sequence, the items of which are the different distinct values of the `region` element, plus a final item, `All`.

**Note:** If the items in the drop-down list are obtained from a page source, then they will appear, by default, in document order.

**Get Value from XPath**
The value returned by the XPath expression is displayed in the control. This enables alternative ways to enter display values for certain controls. For example, a combo box could take its display value from either a page source node or the return value of the *Get Value from XPath* property.

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

**Auto Correct Value**
Possible values are `true` or `false`. If set to `true`, then the value in the associated page source node is corrected to the first XML value in the unsorted list of combo box values. The property's default value is `false`.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node action. The *Auto Correct Value* property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

**Multi Select**
Sets whether the combo box enables multiple values to be selected or not (true/false). The default is `false`.
A combo box normally offers a choice of multiple options, of which only one may be selected. In a MobileTogether solution, the value corresponding to the selected option is written to a page source node.

If the Multi Select property is set to true, then the user may select more than one option. The values corresponding to each of the selected options are concatenated into a single string, and entered in the associated page source node. When the string is constructed, the selected values are separated from each other by a separator that you specify in the Multi Select Separator property (see below). Note that the combo box values themselves must not contain the separator.

In the screenshot below, the values corresponding to the selected options are concatenated into a single string (with comma separators) and entered in the `weekDay` attribute node (highlighted). The separator for the string that is entered in the page source is specified via the Multi Select Separator property (see below), while the separator for the string that is displayed in the combo box itself is specified via the Multi Select Separator Visible property (see below).

![Combo Box Screenshot](image)

**Note:** If, in subsequent processing, multiple values contained in a string need to be separated, then the string must be tokenized using the same separator that was used when constructing the string. You can use the XPath `tokenize` function for this.

- **Multi Select Separator**
  
  This property is enabled when the Multi Select property is set to true. If the end user selects multiple values in the combo box, then a string is constructed from the selections by concatenating the XML values of the selections and inserting the separator between values. The string that is constructed in this way is entered in the associated page source node. Note that a different separator can be used to construct the
Controls and Control Events

The availability of different separators enables an optimal choice of separator for XML processing on the one hand and a human-readable device display on the other.

You can either: (i) select a separator from the available choices, or (ii) enter one or more characters that you want to use as the separator, either directly or via an XPath expression.

**Note**: If you use an XPath expression to return the separator, note that the expression is evaluated when the combo box values are parsed. So, if the expression is such that its return value changes during runtime processing (for example, if it depends on a node with changing content), then the returned value might not be the desired character/s.

Also see the the **Multi Select** property above for related information.

### Multi Select Separator Visible

This property is enabled when the **Multi Select** property is set to **true**. If the end user selects multiple values in the combo box, then a string is constructed from the selections by concatenating the Visible Entry values of the selections and inserting the separator between values. The string that is constructed in this way is displayed in the combo box. Note that a different separator can be used to construct the string that is entered in the associated page source node; see the **Multi Select Separator** property above.

The availability of different separators enables an optimal choice of separator for XML processing on the one hand and a human-readable device display on the other.

You can either: (i) select a separator from the available choices, or (ii) enter one or more characters that you want to use as the separator, either directly or via an XPath expression.

**Note**: If you use an XPath expression to return the separator, note that the expression is evaluated when the combo box values are parsed. So, if the expression is such that its return value changes during runtime processing (for example, if it depends on a node with changing content), then the returned value might not be the desired character/s.

Also see the the **Multi Select** property above for related information.

### Control Action

Click the **Additional Dialog** button to display the control's **Actions dialog**. You can set actions to perform when a control event is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the **Actions dialog**. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

### Visible

An XPath expression that should evaluate to **true()** or **false()**. If the expression evaluates to **false()**—and only if it evaluates to **false()**—then the control is not visible. If the expression evaluates to **true()** or returns some other value, then the control is visible. The default is **true()**. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The **Visible** property can be used
to render an object visible or not depending upon whether an XPath expression evaluates to `true()`.
As a result, the display of an object can be made to be dynamically dependent on the content or structure of
data.

**Note:** For information about the visibility of spanned columns/rows, see [Table Properties](#).

**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. 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’.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- off (the text is not automatically resized; this is the default)
- ellipsis (adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)
- individually (the text of this control only will be automatically resized)
- group X (where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- template group X (where X=1 to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as...
a group if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**
- If the Multiline property has been set to true, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines.
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's Wrap Content Auto-Fit Group property.

▼ **Bold Text**
Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

▼ **Italic Text**
Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

▼ **Underline Text**
Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

▼ **Background Color**
Sets the background color of the object when the object is enabled. You can do one of the following to select the color:
- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control’s background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:**
You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property’s combo box, or (ii) entering the XPath expression "control" in the property’s value field.

**Note:**
A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

▼ **Background Color (Disabled)**
Sets the background color of the object when the object is disabled. You can do one of the following to
select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is enabled, use the Background Color property.

**Tooltip**
Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**
This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**
Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

**Control Width**
Sets the width of the control. Select a value from the property's combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- `percent value`: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- `pixel, dp, or sp value`: select a pixel, dp, or sp value from the dropdown list, or enter a value
In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- Image and Chart: For these, the default is `wrap_content`.
- Geolocation Map: The default is the smaller of the two values `$MT_CanvasX` and `$MT_CanvasY`. These two dynamic variables give, respectively, the width and height of the device’s viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of `$MT_CanvasX` and `$MT_CanvasY`), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see `Sizes: Pixels, DPI, DP, SP`.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Max Control Width**

This property is available only when the control’s `Control Width` property has been set to `wrap_content`. The `Max Control Width` property sets the maximum width of the control. Select a value from the property’s combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

**Margin**

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the
containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices
If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Padding
Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices
If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically
maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping
the values in this way (from viewport values to device values) ensures that design components
maintain the same size relationship to both the canvas and to each other, no matter what the
resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic
variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to
these dimensions. (For iOS devices, the values returned by these variables are calculated as follows:
The pixel dimensions of the current device coordinate space are converted (using an appropriate
conversion factor) to point values in the viewport coordinate space. These point values—as numbers
—are returned by the variables as pixels for use in the design.) For example, if you want an image to
be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5;
the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

**Tab Order**

The Tab Order property takes an integer as its value, or an XPath expression that evaluates to an integer.
This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the Tab
key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page
via the Page | Show/Define Tab Order 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).

### 9.1.6 Date

Date controls are used to format dates obtained from a node in a page source. This is useful, for example, if
you wish to format dates differently for different regions: 12–31–2014 (US format) and 31–12–2014 (EU format).
The formatting is defined in the Date/Time Format String property (see below for details). Note that the
source node content must be in the correct lexical format as defined by the XSD specification: YYYY-MM-DD.
Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

**All Styles**

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
}```
"Background Color" : $XML1/R/@background,
"Text Size"        : $XML1/R/@textsize
}

map{
  "Style Sheet"      : "Sheet-1"
}

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Date/Time Format String

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (screenshot below).

![Format Dialog](image)

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:
• \texttt{xs:date}: 2014-12-31
• \texttt{xs:time}: 23:59:59
• \texttt{xs:dateTime}: 2014-12-31T23:59:59

\subsection*{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 \texttt{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.

\subsection*{Time Handling}
Specifies how to handle the \texttt{Time} part of a value. The three available options are:

- \textit{Cut time}, which removes the \texttt{Time} part
- \textit{Set time to zero}, which keeps the \texttt{Time} part, but changes it to 00:00:00
- \textit{Keep time}, which keeps the \texttt{Time} part

\subsection*{Control Action}
Click the \textbf{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 event/s are predefined and each is shown in its own 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 \textbf{Additional Dialog} button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

\subsection*{Visible}
An XPath expression that should evaluate to \texttt{true()} or \texttt{false()}. If the expression evaluates to \texttt{false()} — and only if it evaluates to \texttt{false()} — then the control is not visible. If the expression evaluates to \texttt{true()} or returns some other value, then the control is visible. The default is \texttt{true()}. Double-click inside the value field, or click the \textbf{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 \texttt{true()}. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

\textbf{Note:} For information about the visibility of spanned columns/rows, see \textbf{Table Properties}.

\textbf{Note:} The \texttt{$MTControlValue} variable is not available for the evaluation of the Visible property. If it is used, then a validation error results.

\subsection*{Enabled/Editable}
The control is either enabled or disabled, according to whether the value of the property is \texttt{true} (enabled) or \texttt{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. 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'.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property's
values are:

- **off** *(the text is not automatically resized; this is the default)*
- **ellipsis** *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- **individually** *(the text of this control only will be automatically resized)*
- **group X** *(where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.*
- **template group X** *(where X=1 to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.*

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**
- If the Multiline property has been set to **true**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines.
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's Wrap Content Auto-Fit Group property.

**Bold Text**

Select **true** or **false** from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is **false**.

**Italic Text**

Select **true** or **false** from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is **false**.

**Underline Text**

Select **true** or **false** from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is **false**.

**Background Color**

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item **Control's background color** in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control’s background color* in the property’s combo box, or (ii) entering the XPath expression "control" in the property’s value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control’s *Enabled/Editable* property. To set a background color for when the object is disabled, use the *Background Color (Disabled)* property.

- **Background Color (Disabled)**
  Sets the background color of the object when the object is disabled. You can do one of the following to select the color:
  
  - Click the color palette to select a background color
  - Select a color from the dropdown list of the combo box
  - Double-click in the value field to enter a color code (for example, #FF0000), or click the *XPath* toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control’s *Enabled/Editable* property. To set a background color for when the object is enabled, use the *Background Color* property.

- **Tooltip**
  Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

- **Hint**
  Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the *XPath* toolbar button and enter an XPath expression to generate the required text.

- **Text Color Hint**
  Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (see above). You can do one of the following to select the color:
  
  - Click the color palette to select a text color
  - Select a color from the dropdown list of the combo box
  - Double-click in the value field to enter a color code (for example, #FF0000), or click the *XPath* toolbar button and enter an XPath expression to generate the required text.
toolbar button and enter an XPath expression to generate the required text.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

**Control Width**

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- *fill_parent*: makes the control as wide as the parent, which could be, for example, a table cell or the page
- *wrap_content*: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- *wrap_content_longest_entry*: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- *percent value*: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- *pixel, dp, or sp value*: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, *fill_parent* creates a maximum width, while *wrap_content* creates a minimum width. If the combo box is within a table cell, for example, *fill_parent* would let the combo box fill the cell whereas *wrap_content* might not fill the cell.

The default value is *fill_parent* for all controls except the following:

- *Image and Chart*: For these, the default is *wrap_content*.
- *Geolocation Map*: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to $\min(MT_CanvasX, MT_CanvasY)$.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

- Points versus pixels on iOS devices
If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px')

Max Control Width

This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.
In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Tab Order

The Tab Order property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.
The tab order is the sequence in which controls receive the focus when the client-device user taps the Tab key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the Page | Show/Define Tab Order 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).

### 9.1.7 DateTime (iOS)

DateTime controls are available only on iOS client devices and are used to format dateTimes obtained from a node in a page source. This is useful, for example, if you wish to format dateTimes differently for different regions: 12-31-2014 12:00:00 (US format) and 31-12-2014 12:00:00 (EU format). The formatting is defined in the Date/Time Format String property (see below for details). Note that the source node content must be in the correct lexical format as defined by the XSD specification: YYYY-MM-DDTHH:MM:SS. Optional timezone and millisecond parts are allowed.

**Notes**

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

▼ All Styles
The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
  "Bold Text"       : $XML1/R/@bold = "1",
  "Italic Text"     : true(),
  "Text"            : "hello",
  "Text Color"      : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size"       : $XML1/R/@textsize
}

map{
  "Style Sheet"     : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example,
**Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the **Styles & Properties Pane**.

- You can also specify a style sheet to use, as shown in the second map above.

**Date/Time Format String**

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (screenshot below).

![Format dialog screenshot](image)

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 event/s are predefined and each is shown in its own tab in the right-hand pane of the **Actions dialog**. A library of actions is displayed in the left-hand
Controls and Control Events

You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's Additional Dialog button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

**Visible**

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

**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. 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 **Text Size** 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`.  

---

© 2015-2021 Altova GmbH
For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values— as numbers —are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Text Size Auto-Fit**

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- **off** *(the text is not automatically resized; this is the default)*
- **ellipsis** *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- **individually** *(the text of this control only will be automatically resized)*
- **group X** *(where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.*
- **template group X** *(where X=1 to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.*

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**

- If the Multiline property has been set to true, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break
characters in the text would, however, create new lines.

- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's Wrap Content Auto-Fit Group property.

**Bold Text**
Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

**Italic Text**
Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

**Underline Text**
Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

**Background Color**
Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

**Background Color (Disabled)**
Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal
Details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is enabled, use the Background Color property.

**Tooltip**

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Hint**

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field to edit the textual hint, or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Text Color Hint**

Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (see above). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiligne (see the Multiline property).

**Control Width**

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- fill_parent: makes the control as wide as the parent, which could be, for example, a table cell or the page
- wrap_content: makes the control only as wide as the control's content requires; when this value
is selected, the property Max Control Width becomes available

- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- `percent value`: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- `pixel, dp, or sp value`: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- Image and Chart: For these, the default is `wrap_content`.
- Geolocation Map: The default is the smaller of the two values `$MT_CanvasX` and `$MT_CanvasY`. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of `$MT_CanvasX` and `$MT_CanvasY`), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see `Sizes: Pixels, DPI, DP, SP`.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Max Control Width**

This property is available only when the control's Control Width property has been set to `wrap_content`. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:
• **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
• **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

### Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Padding

Sets the control’s padding, which is the space between the control’s border and the control’s content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).
Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Style Sheet

The Style Sheet property sets the style sheet to use for the control. 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).

9.1.8 Edit Field

Edit fields can be used to allow input from the end user. If the end-user input is intended to be stored in a page source node, then an association must be made in the design with this node (by dropping the node from the page source onto the control). In this case, ensure that the edit field is editable by setting the Enabled/Editable property to true. The value of the associated node is displayed in the edit field. When the end user modifies the edit field, the associated node is updated automatically.

Instead of associating a node with the edit field, the content of the edit field can be defined by an XPath expression in the Text property (see below). If the XPath expression locates an XML node, the effect is the same as associating a node by dropping it on to the control. The XPath expression can also be used to
calculate an output and display it in the edit field. In this case the Enabled/Editable property will be set to false automatically, and cannot be edited. It then functions in the same way as a Label control.

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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

All Styles

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

map{
  "Bold Text" : $XML1/@bold = "1",
Note the following points:

- When you enter a value for the **All Styles** property—even if it is not a map—all the styling properties of the current component in the [Styles & Properties Pane](#) will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key-value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the [Styles & Properties Pane](#).
- You can also specify a style sheet to use, as shown in the second map above.

### Keyboard

Defines the type of keyboard that is displayed on the client when the user starts to edit content. Select one of the following options from the dropdown list of the combo box, or enter an XPath expression to select/generate the value you want. The default is client-specific

- Text
- Number
- Password: *The password is hidden.*
- Visible Password: *The password is shown.*
- Email
- URI
- Phone

### Text

The **Text** property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the [Rich Text](#) control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the [Rich Text](#) section for more information.
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. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

Note
- If this property is set to true, and if the Text Size Auto-Fit property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the Max Number of Lines property becomes available.
- Check boxes with multilines can be vertically aligned via the Vertical Alignment property.

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

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 \( \text{xs:date} \) (for the Date control), \( \text{xs:time} \) (for the Time control), or \( \text{xs:dateTime} \) (for the Date, Time, and DateTime controls). Basic examples are:

- \( \text{xs:date}: 2014-12-31 \)
- \( \text{xs:time}: 23:59:59 \)
- \( \text{xs:dateTime}: 2014-12-31T23:59:59 \)

**Date/Time Format Language**

Select one of the supported languages from the dropdown list of the combo box (EN, DE, ES, FR, JA). The selected language will be used in the date/time formatting that is set in the Date/Time Format String property (see description above). If the names of months and weekdays are used in the format string, then these will be displayed in the language selected for this property. The default language is English.

**Trigger Control Actions**

Control actions can be triggered as soon as typing starts, during typing (after a specified time interval), or after typing has been completed. For example, the page can be scrolled to bottom (a control action) when typing starts, or a node can be updated (another control action) after typing has been completed.

- The available values of this property are OnTyping, OnFinishEditing, or AfterTimeInterval. The default value is OnFinishEditing for web clients and OnTyping for all other clients.
- You can select the value you want or enter it via an XPath expression (by clicking the XPath toolbar icon).
- If AfterTimeInterval is selected, then the Trigger Control Actions on Typing Interval [ms] property becomes available (see below).
- Editing is considered to be finished when Return is pressed or when the end user taps outside
the edit field.

**Trigger Control Actions on Typing Interval [ms]**

This property becomes available if the Trigger Control Actions property is set to a value of AfterTimeInterval *(see above)*. It specifies the time interval in milliseconds from when typing starts to when the control action/s is/are triggered. The default interval is 500ms.

**Control Action**

Click the Additional Dialog button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's Additional Dialog button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

**Visible**

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false() — and only if it evaluates to false() — then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

*Note:* For information about the visibility of spanned columns/rows, see Table Properties.

*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.
Controls and Control Events

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 enabled, use the Text Color property.

 Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a text color for when the control is enabled, use the Text Color property.
Text Size

Select a size from the dropdown list of the combo box. Allowed values are: smallest | small | medium | large | largest. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value medium.

You can generate other values by using the `mt-font-height` function. For example, to get a size that is 120% larger than the numeric size that corresponds to ‘largest’ on a device, use the following XPath expression for the TextSize value:

```
mt-font-height('largest', 'px') * 1.2
```

The function generates the numeric (pixel) value that corresponds to the ‘largest’ size. This value is then multiplied by 1.2 to obtain the numeric value that is 120% of the value that corresponds to ‘largest’.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Bold Text

Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

Italic Text

Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

Underline Text

Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to
select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

**Background Color (Disabled)**

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is enabled, use the Background Color property.

**Tooltip**

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Hint**

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Text Color Hint**
Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (see above). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

**Control Width**

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the combo box is within a table cell, for example, **fill_parent** would let the combo box fill the cell whereas **wrap_content** might not fill the cell.

The default value is **fill_parent** for all controls except the following:

- **Image and Chart**: For these, the default is **wrap_content**.
- **Geolocation Map**: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to $\min(MT_CanvasX, MT_CanvasY)$. 

© 2015-2021 Altova GmbH

Altova MobileTogether Designer
For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Max Control Width**

This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- **percent value:** a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value:** select a pixel, dp, or sp value from the dropdown list, or enter a value directly

**Margin**

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top, and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left, and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**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
**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`. 

**Padding**

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the **Padding** property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set **Padding** to be 6px and **Padding Bottom** to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

---

© 2015-2021 Altova GmbH

Altova MobileTogether Designer
### 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).

### 9.1.9 Geolocation Map

A Geolocation Map control can be used to display the map of a specific area: in street, satellite, or hybrid view (the view is defined in the Map Type property). Points of interest in the area covered by the map can be shown by markers. Any number of markers can be defined in the control’s Markers property. The current location as well as zoom controls can optionally be included in the map (see their respective properties below). The size and magnification of the map that is initially displayed in the control can be defined via the Viewport, Control Width, and Control Height properties. The Geolocation Map control also has a OnGeoMapMarkerClicked event, which enables you to define actions to perform when a particular marker is clicked by the user.

### Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

### Geolocation Map events

The **OnGeoMapMarkerClicked event** is available, which enables you to define actions to perform when a marker in the map is clicked. In order to retrieve information about the marker that has been clicked, use the **$MT_GeolocationMapMarker** dynamic variable.

The dynamic variable **$MT_GeolocationMapMarker** contains information about the marker that was last clicked by the client's user. This information that is contained in the variable is stored in an XPath-map construct, in a format that is shown by the following example:

```xml
map {
  "id":"vie",
  "geolocation":(48.2143531, 16.3707266),
  "title":"Vienna",
  "text":"Altova EU"
}
```

To fetch a value from the XPath-map construct, use an XPath expression like this:

```xml
map:get( $MT_GeolocationMapMarker, "id" )
```

This particular expression returns the value of the **id** key (that is, the **id** of the marker that was clicked).

For a description of the actions that can be defined for the **OnGeoMapMarkerClicked event**, see the **Actions section**.

### Geolocation Map 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.

#### All Styles

The **All Styles** property becomes visible if the **All Styles** setting of the More Project Settings dialog.
has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
}
map{
    "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key-value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

**Map Type**

Specifies the map type that is displayed in the Geolocation Map control. The options are:

- Default (no selection): Uses the map type that is selected on the client device
- Street: Shows a street plan layout of the displayed
- Satellite: Shows a satellite picture of the displayed area
- Hybrid: Shows a combination of the street and satellite map types

**Note:** An XPath expression can also be used to specify the map type dynamically. The expression must evaluate to one of the following: (i) street, (ii) satellite, (iii) hybrid.

**Note:** On web clients and in the simulator, only a single map type is available no matter which option from those listed above is selected.

**Show Current Location**

Specifies whether the current location of the client device is displayed in the map. This would be useful to help users locate themselves in the map. The values of the property are true or false, with the default being false. To define the property's value dynamically, use an XPath expression (for example: `Element/@attribute="1"` would evaluate to true or false depending on the value of `Element/@attribute`).
Control and Control Events

Note: For this feature to work, GPS permission must be granted on the client device.

Show Zoom Controls
Specifies whether zoom controls are displayed in the map. The values of the property are true or false, with the default being true. You can use an XPath expression to define the property's value dynamically. Note that zoom controls are not displayed on iOS devices.

Markers
An XPath expression that defines the number of markers that are placed in the map, as well as their properties. The XPath expression must resolve to one or more XPath-map constructs: map-1, map-2, ..., map-N. Each XPath-map construct consists of a set of key-value pairs that together define the properties of one marker. The following keys are specified:

- id: The value is an ID for the marker
- geolocation: This is a sequence of two strings or two numbers (integer or decimal): the latitude and the longitude; for example: (48.2143531, 16.3707266)
- title: A string that is the title of the popup that appears when the marker (in the map) is clicked
- text: A string that is the text of the popup that appears when the marker (in the map) is clicked

A full XPath-map construct would look something like this:

```xml
map {
   "id":"vie",
   "geolocation":(48.2143531, 16.3707266),
   "title":"Vienna",
   "text":"Altova EU"
}
```

The XPath-map construct above would create a marker for the Altova EU office in Vienna. To add more markers to the map, enter additional XPath-map constructs with a preceding comma separator: map-1, map-2, ..., map-N.

mt-geo-map-marker() The MobileTogether XPath extension function mt-geo-map-marker can be used to generate a marker dynamically. Each mt-geo-map-marker function returns an XPath-map construct, that is, one marker. To create multiple markers, multiple mt-geo-map-marker functions can be used. For more information about the function, see the description of mt-geo-map-marker().

Viewport
An XPath expression that evaluates to two geolocation coordinates: the top-left corner and the bottom-right corner of the viewport rectangle. The viewport is the geolocation area that is shown in the geolocation map. Note, however, that the dimensions of the Geolocation Map control are defined by the Control Width and Control Height properties. The viewport, if correctly defined, will lie within the map area shown in the control.

The XPath expression must evaluate to a sequence of two strings, where each string is one set of coordinates. In the following example, for instance, the first string is the top-left coordinate while the second string is the bottom-right coordinate: ("61° -10°", "41° 10°"). The top-left coordinate would

© 2015-2021 Altova GmbH

Altova MobileTogether Designer
thus be 61°N 10°W, while the bottom-right coordinate would be 41°N 10°E. These coordinates would create a viewport of roughly 10° square centered on London (approx. 51°N 0°E).

The default setting for the viewport are the coordinates of a rectangle that displays all defined markers at the highest possible zoom level.

**geolocations-bounding-rectangle()**
The Altova extension function *geolocations-bounding-rectangle* can be used to create a bounding rectangle around a set of submitted geolocations. The function returns the top-left corner and bottom-right corner of a rectangle that fits around the submitted geolocations at the greatest possible zoom level, that is, as closely as possible around the geolocations. The return value of the function is a sequence of two strings; so the function can be used in an XPath function to generate the value of the Viewport property.

For more information about the function, see the description of *geolocations-bounding-rectangle()*. 

**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 event/s are predefined and each is shown in its own 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. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

**Visible**
An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()` —and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the *XPath* button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

*Note:* For information about the visibility of spanned columns/rows, see Table Properties.

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

**Tooltip**
Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long press the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

*Note:* Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.
### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multilined (see the Multiline property).

### Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **Fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- **Percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **Pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, fill_parent creates a maximum width, while wrap_content creates a minimum width. If the combo box is within a table cell, for example, fill_parent would let the combo box fill the cell whereas wrap_content might not fill the cell.

The default value is fill_parent for all controls except the following:

- **Image and Chart**: For these, the default is wrap_content.
- **Geolocation Map**: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to min($MT_CanvasX$, $MT_CanvasY$).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

- **Points versus pixels on iOS devices**

  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these
values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a **point** here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be **concat($MT_CanvasX * 0.5, 'px')**.

### Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as high as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as high as the control's content requires.

In effect, **fill_parent** creates a maximum height, while **wrap_content** creates a minimum height.

**Default values**

- For the Geolocation Map control, the default is the smaller of the two values $MT_CanvasX and $MT_CanvasY. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX and $MT_CanvasY), the default shape and size of the viewport in the control will always be a square with side equal to \( \min(\text{MT_CanvasX}, \text{MT_CanvasY}) \).
- For all other controls, the default is **wrap_content**.

### Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the **Margin** property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set **Margin** to be 6px and **Margin Bottom** to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see **Sizes: Pixels, DPI, DP, SP**.
If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a **point** here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Style Sheet

The **Style Sheet** property sets the **style sheet to use for the control**. 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).

### 9.1.10 Horizontal Line

The Horizontal Line control enables you to add lines that you can style for color and width.

### Notes

- To reset a style or property (in the **Styles & Properties Pane**), 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.

▼ All Styles
The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
  "Bold Text" : $XML1/R/@bold = "1",
  "Italic Text" : true(),
  "Text" : "hello",
  "Text Color" : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size" : $XML1/R/@textsize
}
map{
  "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Line Width
Sets the width (thickness) in pixels of the line. Select a width value (in pixels, dp, or sp) from the dropdown list of the combo box, or double-click in the value field to enter a numeric value. You can also use an XPath expression. The numeric value is understood to be a pixel value. For this reason, no unit
should be specified.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a **point** here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Line Style**

Specifies the style of the line. You can select one of the options from the dropdown list of the combo box, or use an XPath expression. The default value is **solid**.

**Line Color**

Specifies the color of the line. You can do one of the following to select the color:

- Click the color palette to select a line color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the **XPath** toolbar button and enter an XPath expression to generate or fetch the required color code

**Visible**

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The **Visible** property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

**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
This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

Vertical Alignment
Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

Control Width
Sets the width of the control. Select a value from the property's combo box. The following values are available:

- fill_parent: makes the control as wide as the parent, which could be, for example, a table cell or the page
- wrap_content: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- wrap_content_longest_entry: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, fill_parent creates a maximum width, while wrap_content creates a minimum width. If the combo box is within a table cell, for example, fill_parent would let the combo box fill the cell whereas wrap_content might not fill the cell.

The default value is fill_parent for all controls except the following:

- Image and Chart: For these, the default is wrap_content.
- Geolocation Map: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to $\min(MT_CanvasX, MT_CanvasY)$.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

- Points versus pixels on iOS devices
  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these
values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX\text{ and }$MT_CanvasY\text{ dynamic variables} to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be $\text{concat} (MT_CanvasX \times 0.5, \text{'px'})$.

### Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX\text{ and }$MT_CanvasY\text{ dynamic variables} to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be $\text{concat} (MT_CanvasX \times 0.5, \text{'px'})$.

### 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).

9.1.11 Horizontal Slider

The Horizontal Slider control enables users to select a value along the slider's scale. The selected value can be entered as the value of a page source node (via a page source link; see Notes below). In the Styles & Properties Pane, 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:

<table>
<thead>
<tr>
<th>Action execution</th>
<th>Trigger Control Actions During Sliding</th>
<th>Slider's Trigger Interval (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actions triggered after sliding completed</td>
<td>false (default)</td>
<td>—</td>
</tr>
<tr>
<td>Actions triggered continuously</td>
<td>true</td>
<td>0</td>
</tr>
<tr>
<td>Actions triggered at interval</td>
<td>true</td>
<td>1 – 10000</td>
</tr>
</tbody>
</table>

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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

▼ **All Styles**

The **All Styles** property becomes visible if the **All Styles** setting of the More Project Settings dialog has been set to **true**. (The default of this setting is **false**.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```map{
  "Bold Text" : $XML1/R/@bold = "1",
  "Italic Text" : true(),
  "Text" : "hello",
  "Text Color" : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size" : $XML1/R/@textsize
}
map{
  "Style Sheet" : "Sheet-1"
}
```

Note the following points:

• When you enter a value for the **All Styles** property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.

• This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key-value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

**Get Value from XPath**

The value returned by the XPath expression is displayed in the control. This enables alternative ways to enter display values for certain controls. For example, a combo box could take its display value from either a page source node or the return value of the Get Value from XPath property.

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

**Auto Correct Value**

Possible values are `true` or `false`. The property's default value is `false`. If set to `true` and the entered value is outside the slider's defined range of values, then the entered value is automatically corrected to a value from the range that is closest to the entered value. (The slider's defined range of values is set with the Slider Minimum Value and Slider Maximum Value properties.) For example, if the property's value is set to `true`, the slider's range of values is defined to be between 0 and 100 (the default range), and the slider's associated page source node contains a value of "somestring", then this value is automatically corrected to 0. If the node contained a value of 200, then its value would be auto-corrected to 100. The auto-correction happens immediately the out-of-range value is detected, for example, when the solution is opened or when the value is entered.

**Slider Minimum Value**

Sets the minimum value of the slider. Allowed values are -10000 to 10000. The default value is 0. Select one of the predefined numbers between 0 and 100, or enter an XPath expression.

**Slider Maximum Value**

Sets the maximum value of the slider. Allowed values are -10000 to 10000. The default value is 100. Select one of the predefined numbers between 0 and 100, or enter an XPath expression.

**Slider Color**

The Slider Color property sets the color of the slider's scale line when the slider is enabled. The default value is client-specific.

**Slider Color (Disabled)**

The Slider Color (Disabled) property sets the color of the slider's scale line and the slider's marker when the slider is disabled. The condition/s for enabling and disabling a slider is set on the slider's Enabled/Editable property. The property's default value is client-specific.

**Slider Thumb Color**

The Slider Thumb Color property sets the color of the slider's marker when the slider is enabled. The default value is client-specific.

**Trigger Control Actions During Sliding**
If set to **OnSliding**, control actions are triggered **during** sliding (either continuously or at specified intervals). When set to **OnSliding**, the property **Trigger Control Actions on Sliding Interval [ms]** *(see below)* appears; you can set the trigger intervals here. If set to **OnFinishEditing**, control actions are triggered **after sliding has finished**. Default is **OnFinishEditing**.

### Trigger Control Actions on Sliding Interval (ms)

Specifies the intervals, in milliseconds, at which control actions are triggered. The range of values is 0 to 10000. The default is 1000ms. Select a value from the dropdown list, or enter a value.

### Control Action

Click the **Additional Dialog** button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's **Additional Dialog** button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The `Visible` property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see [Table Properties](#).

**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.

### 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; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

### Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is enabled, use the Background Color property.

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified
via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

### Control Width

Sets the width of the control. Select a value from the property’s combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page.
- **wrap_content**: makes the control only as wide as the control’s content requires; when this value is selected, the property Max Control Width becomes available.
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available.
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly.
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly.

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the combo box is within a table cell, for example, **fill_parent** would let the combo box fill the cell whereas **wrap_content** might not fill the cell.

The default value is **fill_parent** for all controls except the following:

- **Image** and **Chart**: For these, the default is **wrap_content**.
- **Geolocation Map**: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device’s viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to \( \min(MT_CanvasX, MT_CanvasY) \).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

#### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX$ and $MT_CanvasY$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows:...
The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $\text{concat}(\text{MT}_\text{CanvasX} \times 0.5, \ 'px')$. For iOS devices, the pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $\text{concat}(\text{MT}_\text{CanvasX} \times 0.5, \ 'px')$.

#### Margin
Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the **Margin** property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

#### Points versus pixels on iOS devices
If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $\text{MT}_\text{CanvasX}$ and $\text{MT}_\text{CanvasY}$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $\text{concat}(\text{MT}_\text{CanvasX} \times 0.5, \ 'px')$.

#### Padding
Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the **Padding** property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.
For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Style Sheet

The Style Sheet property sets the style sheet to use for the control. 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).

9.1.12 Image

The Image control inserts an image in the design. The image that is selected for insertion can be an image file referenced by a URL, or it can be a string that is Base64-encoded image data. The Image Source Type property specifies which of the two types the image is: a URL-located file, or a Base64 string. To specify that the image (URL or Base64 string) is taken from a page source node, drop that node onto the image control.

The Image control’s properties are listed below.
Note: If the image source (URL or Base64 string) is changed during simulation or while the solution runs, then the image must be explicitly reloaded with the Reload action. 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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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 (On Click) or a longer press (On Long Click). A sequence of different actions can be specified for each type of click (see screenshots at left and middle below). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions be executed after those of the end-user click; these actions are defined after the On Long Click event (see screenshot below right).

- **On Click**: The action/s to perform when the control is tapped (see screenshot above left).
• **On Long Click**: The action/s to perform when the control is pressed for a longer time than a tap (see screenshot above center).

• **Additional actions**: The action/s to perform after the On Click or On Long Click actions have been executed (see screenshot above right). If no action has been set for the On Click or On Long Click events, then the additional action/s are performed directly on a click or long-click.

You can combine actions for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.

![OnButtonClicked Button2](image)

The screenshot above shows that the On Click and On Long Click events each has a sequence of actions defined for it. An additional MessageBox event is defined after the On Long Click event. This MessageBox event will be executed after the On Click or On Long Click sequence of actions has completed.

**On Enter/Escape**

If the control's On Enter or On Escape check box is selected, then the control's actions are executed when the respective key (Enter or Escape) is tapped. The key-tap (Enter or Escape) serves as an alternative to the On Click event, and will work additionally to the click. The screenshot below shows the On Enter and On Escape check boxes of the Button event. Other controls that provide this option look similar and work similarly.

![OnButtonClicked Button1](image)

This setting can also be accessed via the control's On Enter/Escape property, which is described below.

**Note**: If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an Enter or Escape key-tap are marked with a symbol of the respective key.

**Note**: This feature is available on Web clients and Windows clients, and in simulations of all clients.
Image properties

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

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

- **All Styles**
  The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

  The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

  ```
  map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
  }
  map{
    "Style Sheet" : "Sheet-1"
  }
  ```

  Note the following points:

  - When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
  - This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
  - You can enter the key-value pairs of the map in any order.
  - The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
  - You can also specify a style sheet to use, as shown in the second map above.

- **Image Source**
  The value of the Image Source property references an image in one of the following ways:

  - The URL of a binary image file (PNG, BMP, etc). The value of the property must be a URL. The URL is selected in the Specify File dialog (see description below).
  - An image file represented as a Base64-encoded string. The value of the property must be a Base64-encoded string. An XPath expression supplies the string, which could be entered directly or obtained form an XML node.
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 Load
From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the
server with reference to the *Working Directory* (defined in the MobileTogether Server
settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's
containing folder on the server must be a descendant of the *Working Directory*. See the
section *Location of Project Files* for details. You can also choose whether to allow
untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative
Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a
default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are
  missing on the server, they will be created when the file is saved. This option is relevant only
  when saving; it is absent where the action is restricted to file loading.

- **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).

![Screenshot of Global Resource Folder Alias]

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 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, Load/Save File, Load/Save Image, Load/Save Binary File, Load/Save Text File, Read Folder, and Get File Info.

**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).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the Sessions setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

**Filename is defined by the end user (on the client device)**

- **Default file extension for file saving:** When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter:** The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt, html, xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt', 'html', 'xml'`).

- **Optional Default File:** You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- **Web Message Box:** Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

**Image Source Type**

Sets the type of the image 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`. 
The value of this property can also be entered as an XPath expression. The expression must evaluate to a string that is either `url` or `base64`.

**Username**

This property is enabled if `Image Source Type` is `url`. Sets a user name for user access to the resource. Double-click in the property's value field to edit.

**Password**

This property is enabled if `Image Source Type` is `url`. Sets a password for user access to the resource. Double-click in the property's value field to edit.

**Create Before Load**

In the combo box, select the value you want: `true` or `false`. If `true`, the chart or Base64 image is created before the page loads. If `false`, a page sources action must be used to create the chart or image. The default value is `true`.

**Autorotate**

Specifies whether to rotate the image according to its EXIF orientation information. Values are `true` or `false`, with the default being `false`.

**Embed Image**

This property is enabled if `Image Source Type` is `url`. 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 event/s are predefined and each is shown in its own 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. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

**Visible**

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the **XPath** button, to enter or edit an XPath expression. The **Visible** property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a
result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

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

**Tooltip**
Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**
This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**
Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

**Control Width**
Sets the width of the control. Select a value from the property’s combo box. The following values are available:

- fill_parent: makes the control as wide as the parent, which could be, for example, a table cell or the page
- wrap_content: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- wrap_content_longest_entry: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, fill_parent creates a maximum width, while wrap_content creates a minimum width. If the combo box is within a table cell, for example, fill_parent would let the combo box fill the cell whereas wrap_content might not fill the cell.
The default value is **fill_parent** for all controls except the following:

- **Image and Chart**: For these, the default is **wrap_content**.
- **Geolocation Map**: The default is the smaller of the two values `$MT_CanvasX` and `$MT_CanvasY`. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of `$MT_CanvasX` and `$MT_CanvasY`), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see **Sizes: Pixels, DPI, DP, SP**.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Max Control Width**

This property is available only when the control's `Control Width` property has been set to **wrap_content**. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

**Max Control Height**

The Max Control Height property sets the maximum height of the control. Select a value from the property's combo box or enter a value directly. The height may be given in any of the following units: **pixel**, **dp**, or **sp**. The value must include the length unit, similarly to the entries in the combo box.

**Margin**
Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

On Enter/Escape

Takes one of three values:

- **OnEnter**: Specifies that the actions of this control are executed when the Enter key is tapped.
- **OnEscape**: Specifies that the actions of this control are executed when the Escape key is tapped.
- **None**: No action when either Enter or Escape is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to "OnEnter" or "OnEscape". If more than one control on a page is given the same value (OnEnter or OnEscape), then the first visible and enabled control that has the value is selected when the key is tapped. (See the Visible and Enabled/Editable properties.)

This setting can also be made via the dialog to set the control's OnClicked actions (see the description of the control's events above).

**Note**: If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an Enter or Escape key-tap are marked with a symbol of the respective key.

**Note**: This feature is available on Web clients and Windows clients, and in simulations of all clients.
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).

9.1.13 Label

Labels are used to display information. The information displayed could be static, entered by the designer as the Text property of the label. Or it could be dynamic, obtained at runtime from a node in one of the page sources; this dynamic link is known as a page source link. To specify dynamic content for the label (a page source link), either drag the required data tree node from the Page Sources Pane onto the label, or enter, as the Text property of the label, an XPath expression that retrieves data from a node or calculates an output.

Notes

- To reset a style or property (in the Styles & Properties Pane), 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 (On Click) or a longer press (On Long Click). A sequence of different actions can be specified for each type of click (see screenshots at left and middle below). The sequence that will be executed depends on the type of click that the end user performs. You can also define that additional actions be executed after those of the end-user click; these actions are defined after the On Long Click event (see screenshot below right).
- **On Click**: The action/s to perform when the control is tapped *(see screenshot above left).*
- **On Long Click**: The action/s to perform when the control is pressed for a longer time than a tap *(see screenshot above center).*
- **Additional actions**: The action/s to perform after the On Click or On Long Click actions have been executed *(see screenshot above right).* If no action has been set for the On Click or On Long Click events, then the additional action/s are performed directly on a click or long-click.

You can combine actions for the different click events. The example in the screenshot below shows how this is done for the Button event, but it works in the same way for other controls as well.

The screenshot above shows that the On Click and On Long Click events each has a sequence of actions defined for it. An additional MessageBox event is defined after the On Long Click event. This MessageBox event will be executed after the On Click or On Long Click sequence of actions has completed.

**On Enter/Escape**

If the control's On Enter or On Escape check box is selected, then the control's actions are executed when the respective key (Enter or Escape) is tapped. The key-tap (Enter or Escape) serves as an alternative to the On Click event, and will work additionally to the click. The screenshot below shows the On Enter and On Escape check boxes of the Button event. Other controls that provide this option look similar and work similarly.
This setting can also be accessed via the control's On Enter/Escape property, which is described below.

Note: If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an Enter or Escape key-tap are marked with a symbol of the respective key.

Note: This feature is available on Web clients and Windows clients, and in simulations of all clients.

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

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

▼ All Styles
The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```
map{
  "Bold Text" : $XML1/R/@bold = "1",
  "Italic Text" : true(),
  "Text" : "hello",
  "Text Color" : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size" : $XML1/R/@textsize
}
```

```
map{
  "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key-value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Text
The **Text** property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the **XPath** toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the **Rich Text** control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the **Rich Text** section for more information.

**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. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

**Note**

- If this property is set to true, and if the Text Size Auto-Fit property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the Max Number of Lines property becomes available.
- Check boxes with multilines can be vertically aligned via the Vertical Alignment property.

**Max Number of Lines**

This property become available only when the Multiline property has been set to true. It sets the maximum number of lines that are allowed in the control. The property’s default value is unlimited. This property cannot be set if the Text Size Auto-Fit property has been enabled. You must choose either to set a maximum number of lines or to auto-fit the text.

**Number Format String**

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

### 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 event/s are predefined and each is shown in its own 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. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.
Visible

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

Note: For information about the visibility of spanned columns/rows, see Table Properties.

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 disabled, use the Text Color (Disabled) property.
control is enabled, use the **Text Color** property.

### Text Size

Select a size from the dropdown list of the combo box. Allowed values are: `smallest` | `small` | `medium` | `large` | `largest`. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value `medium`.

You can generate other values by using the `mt-font-height` function. For example, to get a size that is 120% larger than the numeric size that corresponds to `'largest'` on a device, use the following XPath expression for the **TextSize** value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the `'largest'` size. This value is then multiplied by 1.2 to obtain the numeric value that is 120% of the value that corresponds to `'largest'`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- **off** (the text is not automatically resized; this is the default)
- **ellipsis** (adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)
- **individually** (the text of this control only will be automatically resized)
- **group X** (where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
• **template group X** (where $X = 1$ to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A **template group** is different from a **group** (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a **template group** as a **group** if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**
- If the Multiline property has been set to **true**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's Wrap Content Auto-Fit Group property.

**Bold Text**
Select **true** or **false** from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is **false**.

**Italic Text**
Select **true** or **false** from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is **false**.

**Underline Text**
Select **true** or **false** from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is **false**.

**Strikethrough Text**
Select **true** or **false** from the dropdown list of the combo box to determine whether a line is drawn through the text. The value can also be generated dynamically via an XPath expression. Default is **false**.

**Automatic Link Detection**
A setting to specify whether a text fragment that has the pattern of a website-page URL or an email URL will be automatically displayed as a link. If the label text contains URLs (such as www.altova.com) or email addresses (such as altova.user@altova.com), then such text can be set to be automatically displayed as a live link in the label text. Clicking or tapping the link opens, respectively, the targeted website page in a browser or a new email in the device's email app.

This property can take one of the following values:

- **None**: The default value. Automatic link detection is not carried out.
- **Explicit Links**: A text fragment that begins with http://, https://, rtsp://, or mailto: is detected to be a link and is displayed as a live link in the control's text.
- **All Links**: Any text that indicates a website-page URL or email URL is automatically detected to be a link. For example, in addition to the explicit links listed above, the following patterns would
also be detected as links: (i) www.altova.com, (ii) altova.com, (iii) altova.user@altova.com.

**Background Color**

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item *Control's background color* in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting *Control's background color* in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's *Enabled/Editable* property. To set a background color for when the object is disabled, use the *Background Color (Disabled)* property.

**Background Color (Disabled)**

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's *Enabled/Editable* property. To set a background color for when the object is enabled, use the *Background Color* property.

**Tooltip**

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in
other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

**Control Width**

Sets the width of the control. Select a value from the property’s combo box. The following values are available:

- `fill_parent`: makes the control as wide as the parent, which could be, for example, a table cell or the page.
- `wrap_content`: makes the control only as wide as the control's content requires; when this value is selected, the property `Max Control Width` becomes available.
- `wrap_content_longest_entry`: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property `Max Control Width` becomes available.
- `percent value`: a percentage of the page width; select a value from the dropdown list, or enter a value directly.
- `pixel, dp, or sp value`: select a pixel, dp, or sp value from the dropdown list, or enter a value directly.

In effect, `fill_parent` creates a maximum width, while `wrap_content` creates a minimum width. If the combo box is within a table cell, for example, `fill_parent` would let the combo box fill the cell whereas `wrap_content` might not fill the cell.

The default value is `fill_parent` for all controls except the following:

- `Image` and `Chart`: For these, the default is `wrap_content`.
- `Geolocation Map`: The default is the smaller of the two values `$MT_CanvasX` and `$MT_CanvasY`. These two dynamic variables give, respectively, the width and height of the device’s viewport. Since the default of both `Control Height` and `Control Width` are the same (in each case, the smaller of `$MT_CanvasX` and `$MT_CanvasY`), the default shape and size of the viewport in the control will always be a square with side equal to `min($MT_CanvasX, $MT_CanvasY)`.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping
the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

▼ Max Control Width

This property is available only when the control’s Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property’s combo box. The following values are available:

- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate
For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### On Enter/Escape

**Take one of three values:**

- **OnEnter**: Specifies that the actions of this control are executed when the **Enter** key is tapped.
- **OnEscape**: Specifies that the actions of this control are executed when the **Escape** key is tapped.
- **None**: No action when either **Enter** or **Escape** is pressed. This is the default value.

If XPath expressions are used to generate the values, the expressions must evaluate to "OnEnter" or "OnEscape". If more than one control on a page is given the same value (OnEnter or OnEscape), then the...
first visible and enabled control that has the value is selected when the key is tapped. (See the Visible and Enabled/Editable properties.)

This setting can also be made via the dialog to set the control's OnClicked actions (see the description of the control's events above).

**Note:** If you select the Page | Show/Define Tab Order menu command, then controls that have been assigned an Enter or Escape key-tap are marked with a symbol of the respective key.

**Note:** This feature is available on Web clients and Windows clients, and in simulations of all clients.

**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).

### 9.1.14 Placeholder Control

The Placeholder Control enables you to add a control template to a page as well as to a control template itself. See the section Control Templates for information about how to work with control templates and Placeholder Controls.

**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.

**Placeholder Control events**

There is no event associated with the Placeholder Control.

**Placeholder Control properties**

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

- **Control Template**
  Selects the control template that this Placeholder Control instantiates. The dropdown list of the property's value field displays all the control templates that have been defined in the design till this time. Alternatively, you can use an XPath expression to select a control template. Such an XPath expression must evaluate to the name of a control template in the design. Using XPath expressions enables you to select templates conditionally. See [Example Projects](#) for ways to do this.

When you select a control template, its parameters are displayed as sub-properties of the Control Template property (see screenshot below). Enter XPath expressions to select or generate the values of the parameters. The context node of the expressions will be the context node of the Placeholder Control when the control is processed. The context node for the evaluation of parameters and variables is not changed by the Control XPath Context property (see below).

If you select a control template via an XPath expression, then the control template is not selected till
Controls and Control Events

runtime, when the expression is evaluated. Since the control template is not known at design time, no parameters can be displayed in the pane. Instead of a list of parameters, a property named Template Parameters is available. You can enter an XPath expression to generate the values of the expected parameters. The expression must be either an array expression or map expression. If you use an array expression, the parameter values must be provided in the same sequence as the parameter definition order in the control template; additionally any optional parameters must not be omitted. See Example Projects for a sample of such expressions.

Control XPath Context

This property is used to change the context node of XPath expressions that are evaluated in the control template that the Placeholder Control instantiates. The instantiated control template is the template named in the Control Template property (see above).

Note the following points:

- XPath expressions for the values of template parameters are evaluated in the context of the Placeholder Control’s context node—not the changed context node.
- XPath expressions that generate the values of template variables, however, will be evaluated in the context specified by this property (Control XPath Context). If no value is set for this property, then values of variables are also evaluated using the Placeholder Control’s context node.

9.1.15 Radio Button

Radio buttons can be used to constrain the user to select one of a set of specific values. Each radio button is associated with one specific value. A set of radio buttons are grouped into a mutually exclusive set by associating all of them with a single page source node. By “mutually exclusive” is meant that the user is constrained to select only one radio button from the set. The value of the radio button that is selected will be entered into the associated page source node. When you insert a radio button control, you can specify whether the radio button should be to the left or right of the button text, or in the default system position.

For example, in the screenshot below, all three radio buttons are associated with one page source node: an element called Product. This creates a set of mutually exclusive radio buttons for the Product node. Each radio button is given a value through its Checked Values property. The button text is the value of the button's Text property.

When the solution is run, all three radio buttons are initially displayed empty (see screenshot below). When the user selects a radio button, that button's checked value (the product name in our example) is passed to the associated node (the Product element; see the XML Data tree in the screenshot below).
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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

- **All Styles**
  
  The **All Styles** property becomes visible if the **All Styles** setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

  The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```java
map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
}
map{
    "Style Sheet" : "Sheet-1"
}
```

Note the following points:
• When you enter a value for the **All Styles** property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.

• This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.

• You can enter the key–value pairs of the map in any order.

• The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.

• You can also specify a style sheet to use, as shown in the second map above.

### Text

The **Text** property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the XPath toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the **Rich Text** control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the **Rich Text** section for more information.

**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. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

**Note**

- If this property is set to true, and if the **Text Size Auto-Fit** property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the **Max Number of Lines** property becomes available.
- Check boxes with multilines can be vertically aligned via the **Vertical Alignment** property.

### Checked Values

Provides an XML data value for the selected state of the control. The default value is 1. To change the XML data value, click the property’s **Additional Dialog** button. In the Edit Checked Values dialog that appears (screenshot below), enter a value for the checked (selected) state. If the radio button is selected by the end-user, the XML data value will be entered as data in the XML node associated with the control.
Note the following points:

- To specify which node will receive the value, make a page source link from the control to a page source node (by dragging the node on to the control).
- If you enter more than one value for the checked value, then the first value will be used; the others are ignored.
- You can enter a value for the Checked Values property in a style sheet for radio buttons.
- If the Checked Values property is defined in a style sheet, and a control is, as a consequence, assigned values for this property in more than one place, then the most local definition wins. The most local definition is the style defined directly in the control's property. For priority involving a style sheet, see Priority within a Style Sheet and Priority across Style Sheets.

### 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 an XML value. These XML values are defined in the Checked Values property. The control is also associated with a page source node. If the value in the page source node is anything other than the XML value defined for the current status of the control (checked or unchecked), then auto-correction updates the value in the node to the value that has been defined for the current status.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node action. The Auto Correct Value property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

For example, if the checked value is defined to be 1 and the unchecked value to be 0, and if the node's content is "somestring", then, (i) if the control is checked, the node's content will be corrected to 1, (ii) if
the control is unchecked, the node's content will be corrected to 0. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction.

The Auto Correct Value property has two possible values: true or false. If the property is set to true, XML values are automatically corrected. The property's default value is false.

Check Mark Position
Sets the position of the check box relative to the control's text: either to the left or right of the text. The default is the operating system default.

Control Action
Click the Additional Dialog button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

After defining a control's actions, you can access and edit them at any time by clicking the property's Additional Dialog button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

Visible
An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

Note: For information about the visibility of spanned columns/rows, see Table Properties.

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. 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 Text Size value: `mt-font-height('largest', 'px') * 1.2`. The function generates the numeric (pixel) value that corresponds to the ‘largest’ size. This value is then multiplied by 1.2 to obtain the numeric value that is 120% of the value that corresponds to ‘largest’.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices
If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.
In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- **off** *(the text is not automatically resized; this is the default)*
- **ellipsis** *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- **individually** *(the text of this control only will be automatically resized)*
- **group X** *(where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.*
- **template group X** *(where X=1 to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.*

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**

- If the Multiline property has been set to **true**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines..
- This property cannot be enabled if the **Max Number of Lines** label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table’s **Wrap Content Auto-Fit Group** property.

### Bold Text

Select **true** or **false** from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is **false**.

### Italic Text

Select **true** or **false** from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is **false**.
### Underline Text

Select `true` or `false` from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is `false`.

### Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item `Control's background color` in 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. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting `Control's background color` in the property's combo box, or (ii) entering the XPath expression `"control"` in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color for when the object is disabled, use the **Background Color (Disabled)** property.

### Background Color (Disabled)

Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, `#FF0000`), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's `Enabled/Editable` property. To set a background color when the object is enabled, use the **Background Color** property.

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.
Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

Vertical Alignment

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- fill_parent: makes the control as wide as the parent, which could be, for example, a table cell or the page
- wrap_content: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- wrap_content_longest_entry: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, fill_parent creates a maximum width, while wrap_content creates a minimum width. If the combo box is within a table cell, for example, fill_parent would let the combo box fill the cell whereas wrap_content might not fill the cell.

The default value is fill_parent for all controls except the following:

- Image and Chart: For these, the default is wrap_content.
- Geolocation Map: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to \( \min(\text{\$MT_CanvasX}, \text{\$MT_CanvasY}) \).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these
values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a **point** here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be **concat($MT_CanvasX * 0.5, 'px')**.

**Max Control Width**

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The **Max Control Width** property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

**Control Height**

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as high as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as high as the control's content requires.

In effect, **fill_parent** creates a maximum height, while **wrap_content** creates a minimum height.

**Default values**

- For the the Geolocation Map control, the default is the smaller of the two values $MT_CanvasX and $MT_CanvasY. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both **Control Height** and **Control Width** are the same (in each case, the smaller of $MT_CanvasX and $MT_CanvasY), the default shape and size of the viewport in the control will always be a square with side equal to **min($MT_CanvasX, $MT_CanvasY)**.
- For all other controls, the default is **wrap_content**.

**Margin**

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-
click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping
the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the \$MT_CanvasX and \$MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to \$MT_CanvasX * 0.5; the XPath expression for this image width would be \texttt{concat(\$MT_CanvasX * 0.5, 'px')}.

\textbf{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.

\textbf{Note}: The Tab Order feature is available on Web and Windows clients only.

\textbf{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.

\textbf{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).

\section*{9.1.16 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.
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.

- **All Styles**
  The **All Styles** property becomes visible if the **All Styles** setting of the [More Project Settings dialog](#) has been set to true. (The default of this setting is false.)

  The property enables you to set all of the component’s styles via a single XPath map expression, such as the two map expressions below:

  ```xml
  map(
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
  )
  
  map(
    "Style Sheet" : "Sheet-1"
  )
  ```

  Note the following points:

  - When you enter a value for the **All Styles** property—even if it is not a map—all the styling properties of the current component in the **Styles & Properties Pane** will no longer be visible.
  - This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
  - You can enter the key–value pairs of the map in any order.
  - The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular
component are listed under that component in the Styles & Properties Pane.

- You can also specify a style sheet to use, as shown in the second map above.

### Text

The Text property takes as its value one of the following:

- A fixed value text string to be displayed in the control
- An XPath expression that retrieves data from a node in a page source and displays this data in the control

Double-click inside the value field to edit, or click the XPath toolbar button and enter an XPath expression to generate the required text. Alternatively, right-click the property and select the entry method you want from the context menu (fixed-value or XPath).

**Note:** In the Rich Text control, you can enter an XPath expression that is an HTML-encoded string. If styles for HTML elements have been defined in the active Rich Text style sheet, then the text value of this property will be displayed with the appropriate formatting. See the Rich Text section for more information.

**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.

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.
Note: Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Hint**

Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Text Color Hint**

Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (see above). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, \#FF0000), or click the **XPath** toolbar button and enter an XPath expression to generate the required text.

**Rich Text Style Sheet**

Specifies which of the design's Rich Text style sheets is to be used for the layout of text in the Rich Text control. The dropdown list of the property's combo box displays all the project's Rich Text style sheets (created via the Rich Text Style Sheets dialog).

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to **left**, **center**, or **right**. Default is **left** for all controls except vertical lines, for which it is **center**. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**

Sets the vertical alignment to **top**, **middle**, or **bottom**. Default is **middle**. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the **Multiline** property).

**Control Width**

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property **Max Control Width** becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property **Max Control Width** becomes available
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a
value directly

- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the combo box is within a table cell, for example, **fill_parent** would let the combo box fill the cell whereas **wrap_content** might not fill the cell.

The default value is **fill_parent** for all controls except the following:

- **Image** and **Chart**: For these, the default is **wrap_content**.
- **Geolocation Map**: The default is the smaller of the two values **$MT_CanvasX** and **$MT_CanvasY**. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both **Control Height** and **Control Width** are the same (in each case, the smaller of **$MT_CanvasX** and **$MT_CanvasY**), the default shape and size of the viewport in the control will always be a square with side equal to **min($MT_CanvasX, $MT_CanvasY)**.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a **point** here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the viewport coordinate space to **pixels** in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the **$MT_CanvasX** and **$MT_CanvasY** dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to **$MT_CanvasX * 0.5**; the XPath expression for this image width would be **concat($MT_CanvasX * 0.5, 'px')**.

### Max Control Width

This property is available only when the control's **Control Width** property has been set to **wrap_content**. The **Max Control Width** property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly
Rich Text Height

Specifies the height of the Rich Text control. The default behavior (wrap_content) is for the height of the control to increase as content increases, up to a maximum of the screen height on devices and browser window on web clients. When the maximum height is reached, the content part of the control displays a scrollbar; the toolbar part is fixed at the top of the control. Alternatively, you can set a fixed height (in pixels, dp, or sp) for the control.

Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Tab Order

The Tab Order property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the Tab key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the Page | Show/Define Tab Order 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).

---

### 9.1.17 Signature Field

The Signature Field control enables the signature of an end user to be stored as a graphic file. This is useful, for example, in courier-business solutions, where signatures are used to acknowledge receipt of a couriered item. As the end user starts drawing his signature in the signature field, the signature is written in Base64 image encoding to a page source node. When the page source is saved, the Base64 image data is saved in it, in the node designated for it.

The signature image has the following default properties. Its background-color is the inverse of the page background-color. The signature itself is the same color as the page background-color. The image width is the smaller of the device's viewport dimensions. The image height is half of its width. How these values are calculated with XPath expressions is shown in the table below, together with the control properties you can use to customize these settings.

<table>
<thead>
<tr>
<th>Signature property</th>
<th>Default value</th>
<th>Custom value via control property...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature color</td>
<td><code>$MT_PageBackgroundColor</code></td>
<td>Text Color</td>
</tr>
<tr>
<td>Signature background-color</td>
<td><code>mt-invert-color($MT_PageBackgroundColor)</code></td>
<td>Background Color</td>
</tr>
<tr>
<td>Signature image width</td>
<td><code>min($MT_CanvasX, $MT_CanvasY)</code>, <code>min($MT_CanvasX, $MT_CanvasY)</code></td>
<td>Signature Creation Width</td>
</tr>
<tr>
<td>Signature image height</td>
<td><code>min($MT_CanvasX, $MT_CanvasY) div 2</code></td>
<td>Signature Creation Height</td>
</tr>
</tbody>
</table>

The main settings for the signature field are:

- a page source link, which is the page source node where the signature image data is stored. Drag a page source node onto the control to create/modify the control's page source link. Delete the page source link to clear the association (see Notes below).
- the Signature Creation Width and Signature Creation Height properties; these specify the dimensions of the image that will be created
The 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.

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 Field 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.

- **All Styles**
  The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)
The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
  "Bold Text" : $XML1/R/@bold = "1",
  "Italic Text" : true(),
  "Text" : "hello",
  "Text Color" : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size" : $XML1/R/@textsize
}
map{
  "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

### Visible

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

**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; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

Note: You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

 Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

Note: Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

Image Anti-Aliasing

Defines whether anti-aliasing is used when the image associated with the control is created. (Anti-aliasing is a technique for removing the jagged edges in images.) The property can take a value of true or false. The default is false. Tip: If image colors are going to be converted subsequently, it is advisable to set the property's value to false. This is because anti-aliasing color information in the original image cannot
reliably be converted to suitable anti-aliasing colors in the target color scheme. Note: For web clients, the value of this property is ignored, and anti-aliasing is always in effect.

**Signature Creation Width**

A number, in pixels, dp, or sp, that specifies the width of the signature image. It is the width of the canvas on which the signature is drawn. For example, a value of 400 will assign a width of 400 pixels to the signature image. Note that this is the width of the image that is created, and is not necessarily that of the image rendered on the mobile device. The rendered image is scaled to the width defined in the **Control Width** property. For example, if the signature image has a width of 400 pixels, and the **Control Width** property has a value of 80%, then, on a device that has a width of 1000 pixels, the image will scale up to a width of 800 pixels (80% of the device width).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Signature Creation Height**

A number, in pixels, dp, or sp, that specifies the height of the signature image. It is the height of the canvas on which the signature is drawn. For example, a value of 200 will assign a height of 200 pixels to the signature image. Note that this is the height of the image that is created; it is not necessarily the height of the image rendered on the mobile device. The height of the rendered image is scaled proportionally to the width defined in the **Control Width** property. For example, if the signature image has a height of 300 pixels and a width of 400 pixels, and the **Control Width** property has a value of 80%, then, on a device that has a width of 1000 pixels, the image will scale up to a width of 800 pixels (80% of the device width) and to a height of 600 pixels (which maintains the image's aspect ratio).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these
values are read as **points** in the *viewport coordinate space*. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the *viewport coordinate space* to **pixels** in the *device coordinate space*. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX$` and `$MT_CanvasY$` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current *device coordinate space* are converted (using an appropriate conversion factor) to **point values** in the *viewport coordinate space*. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5$`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to **left**, **center**, or **right**. Default is **left** for all controls except vertical lines, for which it is **center**. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

### Vertical Alignment

Sets the vertical alignment to **top**, **middle**, or **bottom**. Default is **middle**. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the **Multiline** property).

### Control Width

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property **Max Control Width** becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property **Max Control Width** becomes available
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the combo box is within a table cell, for example, **fill_parent** would let the combo box fill the cell whereas **wrap_content** might not fill the cell.
The default value is fill_parent for all controls except the following:

- Image and Chart: For these, the default is wrap_content.
- Geolocation Map: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to \( \min(MT_CanvasX, MT_CanvasY) \).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX$ and $MT_CanvasY$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values— as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be \( \text{concat}(MT_CanvasX \times 0.5, 'px') \).

Max Control Width

This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

Control Height

Sets the height of the control. Select a value from the property's combo box. The following values are available:

- fill_parent: makes the control as high as the parent, which could be, for example, a table cell or the page
• **wrap_content**: makes the control only as high as the control’s content requires.

In effect, **fill_parent** creates a maximum height, while **wrap_content** creates a minimum height.

**Default values**

• For the Geolocation Map control, the default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device’s viewport. Since the default of both **Control Height** and **Control Width** are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to $\min(MT_CanvasX, MT_CanvasY)$.

• For all other controls, the default is **wrap_content**.

**Margin**

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the **Margin** property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set **Margin** to be 6px and **Margin Bottom** to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a **point** here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX$ and $MT_CanvasY$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Style Sheet**

The **Style Sheet** property sets the **style sheet to use for the control**. 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.

9.1.18 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.

All Styles

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```map{
"Bold Text" : $XML1/R/@bold = "1",
"Italic Text" : true(),
"Text" : "hello",
"Text Color" : "red",
"Background Color" : $XML1/R/@background,
"Text Size" : $XML1/R/@textsize
}
```

```map{
```
Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

**Space Height**

Sets the height in pixels, dp, or sp of the selected space object. Select a height value (in pixels, dp, or sp) from the dropdown list of the combo box, or double-click in the value field to enter a numeric value, or use an XPath expression. The numeric value is understood to be a pixel value. For this reason, no unit should be specified.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

**Visible**

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of
data.

**Note:** For information about the visibility of spanned columns/rows, see [Table Properties](#).

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

### Tooltip

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

### 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).

### 9.1.19  Switch

A switch is associated with a page source node. The end user can choose between two states: `selected` or `unselected`. Depending on which state is chosen, a corresponding XML value is entered as content of the associated node. The values corresponding to the respective states (selected or unselected) are defined in the `Checked Values (true/false)` property (see property description below). A typical use case would be: first, let the end-user's switch selection determine the content of a node; second, let the content of this node determine a page action, such as, whether a control is enabled/editable.

#### Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in [Page Design View](#)) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the
associated control.

- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

- All Styles

  The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

  The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

  ```
  map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
  }
  map{
    "Style Sheet" : "Sheet-1"
  }
  ```
Note the following points:

- When you enter a value for the `All Styles` property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, `Bold Text` and `Text Size` are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

**Multiline**

Sets multiline input/display on or off (true/false). The default is false. If the text of the control is longer than a single line, and the value is true, then the text will wrap to a new line; if the value is false, then the text will be truncated at the end of the first line.

**Note**

- If this property is set to true, and if the `Text Size Auto-Fit` property is enabled, then the text will not wrap to multiple lines; line break characters in the text would, however, create new lines.
- If this property is set to true on a label control, the `Max Number of Lines` property becomes available.
- Check boxes with multilines can be vertically aligned via the `Vertical Alignment` property.

**Checked/Unchecked Values**

Provides an XML data value for the selected/unselected state of the control. The defaults are, respectively, 1 and 0. To change the XML values that will be entered in the associated page source node, click the property's `Additional Dialog` button. In the Edit Checked Values dialog that appears (screenshot below), enter a value for each the checked (selected) and unchecked (unselected) state. When the end user changes the control's state, the value corresponding to the new state (selected/unselected) will be entered in the page source node.
Note the following points:

- To specify which node will receive the value, make a page source link from the control to a page source node (by dragging the node on to the control).
- If you enter more than one value for the *Checked* value, then the first value will be used; the others are ignored.
- If you enter a *Checked* value but do not enter an *Unchecked* value, then the *Unchecked* value is no longer the default value but an empty string.
- You can enter values for the *Checked/Unchecked Values (true/false)* property in a style sheet for: (i) all check boxes, and/or (ii) all switches, and/or (iii) all controls that use *Checked/Unchecked Values (true/false)* (which are check boxes and switches).
- If values for *Checked/Unchecked Values (true/false)* are defined in a style sheet, and a control is, as a consequence, assigned values for this property in more than one place, then the most local definition wins. The most local definition is the style defined directly in the control's property. For priority involving a style sheet, see *Priority within a Style Sheet* and *Priority across Style Sheets*.

### 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 an XML value. These XML values are defined in the Checked Values property. The control is also associated with a page source node. If the value in the page source node is anything other than the XML value defined for the current status of the control (checked or unchecked), then auto-correction updates the value in the node to
the value that has been defined for the current status.

The page source node might come to contain an invalid value if it were updated via some mechanism other than the control; for example, via an Update Node action. The Auto Correct Value property ensures that no undefined value is accepted. Auto-correction happens immediately the undefined value is detected, for example, when the solution is opened or when the incorrect value is entered.

For example, if the checked value is defined to be 1 and the unchecked value to be 0, and if the node's content is "somestring", then, (i) if the control is checked, the node's content will be corrected to 1, (ii) if the control is unchecked, the node's content will be corrected to 0. If there is more than one XML value assigned to the checked state, then the first XML value is used for the correction.

The Auto Correct Value property has two possible values: true or false. If the property is set to true, XML values are automatically corrected. The property's default value is false.

- **Toggle On Text**
  Sets the text that appears in the control when the control is toggled on. Double-click inside the value field to edit.

- **Toggle Off Text**
  Sets the text that appears in the control when the control is toggled off. Double-click inside the value field to edit.

- **Control Action**
  Click the Additional Dialog button to display the control's Actions dialog. You can set actions to perform when a control event is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the Actions dialog. A library of actions is displayed in the left-hand pane. You can drag an action from the left-hand pane into an event's tab and then define the properties of the action. For each event, multiple actions can be set up. They will be executed in the order in which they occur, from top to bottom.

  After defining a control's actions, you can access and edit them at any time by clicking the property's Additional Dialog button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

- **Visible**
  An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

  **Note:** For information about the visibility of spanned columns/rows, see Table Properties.

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

Enabled/Editable

The control is either enabled or disabled, according to whether the value of the property is true (enabled) or false (disabled). The value can be entered directly (by selecting it in the combo box or by double-clicking in the value field and entering the value you want). The value can also be an XPath expression that evaluates to boolean true or false. The default value is true. Typically the XPath expression of the Enabled/Editable property of a control would check the presence or value of a node. If the expression evaluates to true, the control will be enabled.

Text Color

Sets the color of the control's text when the control is enabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a text color for when the control is disabled, use the Text Color (Disabled) property.

Text Color (Disabled)

Sets the color of the control's text when the control is disabled. You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a text color for when the control is enabled, use the Text Color property.

Text Size

Select a size from the dropdown list of the combo box. Allowed values are: smallest | small | medium | large | largest. Each platform/device has its own pixel-height for each size. So the default text size in pixels is the client-specific pixel-height that corresponds to the value medium.

You can generate other values by using the mt-font-height 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'.

© 2015-2021 Altova GmbH
For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

¬ Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- off *(the text is not automatically resized; this is the default)*
- ellipsis *(adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)*
- individually *(the text of this control only will be automatically resized)*
- group X *(where X=1 to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.*
- template group X *(where X=1 to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.*

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

Note

- If the Multiline property has been set to true, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines.
• This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
• In tables, this property can be used with the table's Wrap Content Auto-Fit Group property.

▼ Bold Text
Select true or false from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is false.

▼ Italic Text
Select true or false from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is false.

▼ Underline Text
Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

▼ Background Color
Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

• Click the color palette to select a background color
• Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

Note: You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

▼ Background Color (Disabled)
Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

• Click the color palette to select a background color
• Select a color from the dropdown list of the combo box
• Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of
the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control’s Enabled/Editable property. To set a background color for when the object is enabled, use the **Background Color** property.

**Tooltip**

Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to **left**, **center**, or **right**. Default is **left** for all controls except vertical lines, for which it is **center**. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**

Sets the vertical alignment to **top**, **middle**, or **bottom**. Default is **middle**. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multilie (see the **Multiline** property)

**Control Width**

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control’s content requires; when this value is selected, the property Max Control Width becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, **fill_parent** creates a maximum width, while **wrap_content** creates a minimum width. If the combo box is within a table cell, for example, **fill_parent** would let the combo box fill the cell whereas **wrap_content** might not fill the cell.

The default value is **fill_parent** for all controls except the following:

- **Image and Chart**: For these, the default is **wrap_content**.
- **Geolocation Map:** The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to \( \min(MT_CanvasX, MT_CanvasY) \).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

- **Points versus pixels on iOS devices**

  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

  In MobileTogether Designer, you can use the $MT_CanvasX$ and $MT_CanvasY$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5$; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

- **Max Control Width**

  This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

  - **percent value:** a percentage of the page width; select a value from the dropdown list, or enter a value directly
  - **pixel, dp, or sp value:** select a pixel, dp, or sp value from the dropdown list, or enter a value directly

- **Margin**

  Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

  For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-
independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the *viewport coordinate space*. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the *viewport coordinate space* to **pixels** in the *device coordinate space*. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The *pixel* dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to **point values** in the *viewport coordinate space*. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Padding**

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the **Padding** property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the *viewport coordinate space*. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a *point* here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the *viewport coordinate space* to **pixels** in the *device coordinate space*. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows:...
The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $\text{MT}_\text{CanvasX} \times 0.5$; the XPath expression for this image width would be `concat($\text{MT}_\text{CanvasX} \times 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 dial**) to assign properties specifically for this control. You can also use an XPath expression to generate or fetch the names of the class(es).

### 9.1.20 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 generated for repeating tables as well as for tables with dynamic rows.
Tables, row and column counts can be static or repeating. For repeating tables, rows or columns you have to assign an XML element or define an XPath expression.

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

### Columns
- **Static number of columns:** 2
- **Dynamic number of columns:**
  - Leading columns: 0
  - Repeating columns: 1 (for every element occurrence, this amount of columns will be created)
  - Trailing columns: 0

### Rows
- **Static number of rows:** 2
- **Dynamic number of rows:**
  - Header rows: 0
  - Repeating rows: 1 (for every element occurrence, this amount of rows will be created)
  - Footer rows: 0

- **Automatic Append/Delete controls (repeating table or rows)**

[OK] [Cancel]
Repeating tables and tables with dynamic rows can also have Append/Delete controls automatically added. If added, each instance of a repeating element will have a Delete control next to it. This allows the end user to delete this instance of the element. Depending on whether each instance of the repeating element is created as a table (in repeating tables) or a row (in tables with dynamic rows), the Add control enables a new table or row (and hence an instance of the corresponding element) to be added.

The cells of the table (both static and dynamic) can contain the following:

- Static text
- A node from a page source
- A page control
- A nested table

Table formatting properties are available in the Styles & Properties Pane 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
- All Styles

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
}
map{
    "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.
Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

Note: You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

Skip wrap_content

This property is available for table cells, and takes a value of true or false, with the default being false. If set to true, this cell's content will be ignored when making the wrap_content calculation to determine the containing column's minimum width.

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 page source, is the parent of the day element). Since the week element repeats, a new table will be created for each week element. If, in the page source, there are multiple day child elements of the week element, and if, in the design, the dynamic columns of the column group are not spanned, then the table (for each week) generated from this design will have as many columns as there are day child elements. If, however, you set the Spans Column Groups 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.

### Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top, and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

#### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

#### Padding Left

Sets the control's left padding, which is the space between the control's left border and the control's
content. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

- **Points versus pixels on iOS devices**

  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

  In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, ’px’).

- **Padding Right**

  Sets the control's right padding, which is the space between the control's right border and the control's content. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value.

  For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.
be half the width of the viewport, then give it a width in pixels that is equal to $\text{MT}_\text{CanvasX} \times 0.5$; the XPath expression for this image width would be `concat($\text{MT}_\text{CanvasX} \times 0.5, \ 'px')`.

### Padding Top

Sets the control's top padding, which is the space between the control's top border and the control's content. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

#### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $\text{MT}_\text{CanvasX}$ and $\text{MT}_\text{CanvasY}$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $\text{MT}_\text{CanvasX} \times 0.5$; the XPath expression for this image width would be `concat($\text{MT}_\text{CanvasX} \times 0.5, \ 'px')`.

### Padding Bottom

Sets the control's bottom padding, which is the space between the control's bottom border and the control's content. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

#### Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.
In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be \text{concat}($MT_CanvasX * 0.5, 'px').

**Border Width**

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the Border Width property to display the individual border width properties (left, right, top and bottom), and set the different value. For example: If you set Border Width to be 1px and Border Width Top to be 2px, then the widths of the left, right and bottom borders will all be 1px, while the top border width will be 2px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be \text{concat}($MT_CanvasX * 0.5, 'px').

**Border Color**

Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the Border Color property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set Border Color to be blue and Border Color Top to be red, then the color of the left, right and bottom borders will all be blue, while the top border color will be red.

**Border Style**
Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four sides, expand the Border Style property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set Border Style to be dashed and Border Style Top to be solid, then the style of the left, right and bottom borders will all be dashed, while the top border style will be solid.

**Browser CSS Class**

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the Browser Settings dialog) 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**

**All Styles**

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
    "Text Color" : "red",
    "Background Color" : $XML1/R/@background,
    "Text Size" : $XML1/R/@textsize
}
map{
    "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

**Width**

Sets the width of the object. Select fill_parent or wrap_content or a percentage value from the dropdown list of the combo box. Percentage values are percentages of the parent object.
- **fill_parent**: the width is as wide as the parent, which could be, for example, a table row
- **wrap_content**: the width is only as wide as required by the content; when this value is selected, the property Max Width becomes available. If you want to specify that a particular cell's width not be considered for the calculation of a table-column's width, then set the Skip wrap_content property of the relevant cell to true.
- **wrap_content_fit_parent**: if a table is broader than its parent object, then this value can be used on a column to reduce the column's width. If the table is less wide than its parent object, this value has the same effect as wrap_content.
- **percent value**: a percentage of the parent object; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel value, dp value, or sp value from the dropdown list, or enter a value directly

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Max Width**

This property is available only when the table column's Width property has been set to wrap_content. The Max Width property sets the maximum width of table columns. Select a value from the property's combo box. The following values are available:

- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

**Visible**

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value
field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

Note: For information about the visibility of spanned columns/rows, see Table Properties.

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; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

Note: You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

Note: A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

**Border Width**

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the Border Width property to display the individual border width properties (left, right, top and bottom), and set the different value. For example: If you set Border Width to be 1px and Border Width Top to be 2px, then the widths of the left, right and bottom borders will all be 1px, while the top border width will be 2px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the
resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be \texttt{concat($MT_CanvasX \cdot 0.5, \text{"px"})}.

\subsection*{Border Color}

Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the \texttt{Border Color} property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set \texttt{Border Color} to be \texttt{blue} and \texttt{Border Color Top} to be \texttt{red}, then the color of the left, right and bottom borders will all be \texttt{blue}, while the top border color will be \texttt{red}.

\subsection*{Border Style}

Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four sides, expand the \texttt{Border Style} property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set \texttt{Border Style} to be \texttt{dashed} and \texttt{Border Style Top} to be \texttt{solid}, then the style of the left, right and bottom borders will all be \texttt{dashed}, while the top border style will be \texttt{solid}.

\subsection*{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: \texttt{LabelClassOne LabelClassTwo}. These classes can then be used in a CSS file (specified in the \texttt{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).

\subsection*{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.

\textit{Table Row}

\subsection*{All Styles}

The \texttt{All Styles} property becomes visible if the \texttt{All Styles} setting of the \texttt{More Project Settings dialog} has been set to \texttt{true}. (The default of this setting is \texttt{false}.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

\begin{verbatim}
map{
    "Bold Text" : $XML1/R/@bold = "1",
}
\end{verbatim}
Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

**Visible**

An XPath expression that should evaluate to `true()` or `false()`. If the expression evaluates to `false()`—and only if it evaluates to `false()`—then the control is not visible. If the expression evaluates to `true()` or returns some other value, then the control is visible. The default is `true()`. Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to `true()`. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

**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; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: `"control"`.

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii)
entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the **Background Color (Disabled)** property.

**Border Width**

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the **Border Width** property to display the individual border width properties (left, right, top and bottom), and set the different value. For example: If you set **Border Width** to be 1px and **Border Width Top** to be 2px, then the widths of the left, right and bottom borders will all be 1px, while the top border width will be 2px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

**Border Color**

Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the **Border Color** property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set **Border Color** to be blue and **Border Color Top** to be red, then the color of the left, right and bottom borders will all be blue, while the top border color will be red.

**Border Style**

Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four
sides, expand the *Border Style* property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set *Border Style* to be dashed and *Border Style Top* to be solid, then the style of the left, right and bottom borders will all be dashed, while the top border style will be solid.

**Browser CSS Class**

The name of one or more CSS classes that you want to associate with this control. Use a space to assign multiple classes: `LabelClassOne LabelClassTwo`. These classes can then be used in a CSS file (specified in the [Browser Settings dialog](#)) 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 (i) tables with a repeating table structure, and (ii) rows in a table with a repeating row structure. The *Append* button *(see screenshot below)* appends another occurrence of the repeating structure. The *Delete* button is available for each occurrence of a repeating structure, and deletes that occurrence. The screenshot below shows these buttons in the repeating table structure of a solution.

![Repeating Table Structure](image)

This property defines whether the *Append* and *Delete* buttons are automatically added to the design. The property can have values of `true` and `false`. The default is `true`.

**Create For Each Item In**

This property is available for tables, row groups, and column groups. It sets the number of times the item (table, row, or column) repeats to the number of items returned by the property’s XPath expression.

The XPath expression can select a repeating node in a data tree. This is useful if you want the table, row, or column to be generated as many times as a certain node occurs and want to set the number of these repeats dynamically. In this way, if the number of the selected node's occurrences changes, then the
number of repeats is automatically modified. For example, if the property's XPath expression is $XML1/Office/Department, then the number of times that the table, row, or column is generated will be equal to the number of $XML1/Office/Department elements.

The XPath expression can also be unrelated to a data tree. In this event, the table is created once for each item in the returned sequence. For example:

- 9 times for the expression 1 to 9
- 2 times for "John", "Mary"
- 2 times for 45, true()

Visible

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false() — and only if it evaluates to false() — then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

Note: For information about the visibility of spanned columns/rows, see Table Properties.

Note: The $MTControlValue variable is not available for the evaluation of the Visible property. If it is used, then a validation error results.
to render an object visible or not depending upon whether an XPath expression evaluates to \texttt{true()}. As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

\textbf{Note:} For information about the visibility of spanned columns/rows, see \textit{Table Properties} \footnote{55}.

\textbf{Note:} The \texttt{\$MTControlValue} variable is \textbf{not} available for the evaluation of the \texttt{Visible} property. If it is used, then a validation error results.

\textit{Table}

\textbf{Name}

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

\textbf{All Styles}

The \texttt{All Styles} property becomes visible if the \textit{All Styles} setting of the \textit{More Project Settings} dialog \footnote{160} has been set to \texttt{true}. (The default of this setting is \texttt{false}.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

\begin{verbatim}
map{
  "Bold Text"        : $XML1/R/@bold = "1",
  "Italic Text"      : \texttt{true()},
  "Text"             : "hello",
  "Text Color"       : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size"        : $XML1/R/@textsize
}

map{
  "Style Sheet"      : "Sheet-1"
}
\end{verbatim}

Note the following points:

\begin{itemize}
  \item When you enter a value for the \textit{All Styles} property—even if it is not a map—all the styling properties of the current component in the \textit{Styles & Properties Pane} \footnote{138} will no longer be visible.
  \item This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
  \item You can enter the key–value pairs of the map in any order.
  \item The key names are the names of style properties (or styles). In the first map above, for example, \texttt{Bold Text} and \texttt{Text Size} are style names. The styles that are available for a particular component are listed under that component in the \textit{Styles & Properties Pane} \footnote{138}.
  \item You can also specify a style sheet to use, as shown in the second map above.
\end{itemize}

\textbf{Repeating}

Defines whether the table is \texttt{repeating} \footnote{119} or \texttt{static} \footnote{119}. Its values are \texttt{true} or \texttt{false}. The value of this property is automatically assigned at the time a repeating (=\texttt{true}) or static table (=\texttt{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 (i) tables with a repeating table structure, and (ii) rows in a table with a repeating row structure. The Append button (see screenshot below) appends another occurrence of the repeating structure. The Delete button is available for each occurrence of a repeating structure, and deletes that occurrence. The screenshot below shows these buttons in the repeating table structure of a solution.

![Repeating Table Structure](image)

This property defines whether the Append and Delete buttons are automatically added to the design. The property can have values of true and false. The default is true.

### Create For Each Item In

This property is available for tables, row groups, and column groups. It sets the number of times the item (table, row, or column) repeats to the number of items returned by the property's XPath expression.

The XPath expression can select a repeating node in a data tree. This is useful if you want the table, row, or column to be generated as many times as a certain node occurs and want to set the number of these repeats dynamically. In this way, if the number of the selected node's occurrences changes, then the number of repeats is automatically modified. For example, if the property's XPath expression is $XML1/Office/Department, then the number of times that the table, row, or column is generated will be equal to the number of $XML1/Office/Department elements.

The XPath expression can also be unrelated to a data tree. In this event, the table is created once for each item in the returned sequence. For example:

- 9 times for the expression 1 to 9
- 2 times for "John", "Mary"
- 2 times for 45, true()

### Max Table Width

The Max Table Width property specifies the table's width: (i) in pixels, dp, or sp, (ii) relative to the
device's screen width, or (iii) optimized for columns (\textit{wrap\_content}). The default is \textit{wrap\_content}. Select the value you want from the dropdown list of the property's combo box. If the table width is larger than the screen width, then the table will be displayed with a horizontal scroll bar.

For information about the relationships between pixels, \(\text{dp}\) (device-independent pixels), and \(\text{sp}\) (scale-independent pixels), see \textit{Sizes: Pixels, DPI, DP, SP}.

\textbf{Points versus pixels on iOS devices}

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the \$MT\_CanvasX and \$MT\_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to \$MT\_CanvasX \(* 0.5\); the XPath expression for this image width would be \texttt{concat($MT\_CanvasX \* 0.5, 'px')}.

For more information, see \textit{Table Properties}.

\textbf{Max Table Height}

The \textbf{Max Table Height} property specifies the table's height on the device's screen, in pixels, \(\text{dp}\), or \(\text{sp}\), or relative to the device's screen height. Select one of the values from the property's combo box. For example, if you select 50\%, then the table will have a height of half of the device's screen height. If the table has a vertical extent that does not fit in the allotted screen space, then the table will have a vertical scroll bar and the end user can scroll the rest of the table in and out of the allotted screen space (in this example, 50\% of the screen height). If design components occur above the table, then all these components are displayed above the table; the table itself will have the absolute or relative height specified via this property.

Besides pixel values and percentages of the screen height, the \textbf{Max Table Height} property can take two other values:

- \textit{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.
- \textit{Rest of screen height (always)}: This option enables you to use the entire screen height to display the page. If a table does not have enough vertical extent to fill the page, then additional space is added below the table so that the last component of the page is displayed just above the bottom of the screen.

For information about the relationships between pixels, \(\text{dp}\) (device-independent pixels), and \(\text{sp}\) (scale-independent pixels), see \textit{Sizes: Pixels, DPI, DP, SP}.
**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as **points** in the **viewport coordinate space**. The **viewport coordinate space** is the canvas on which the design components are drawn, and a **point** is the length unit in this space; a **point** here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the **points** of the **viewport coordinate space** to **pixels** in the **device coordinate space**. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX` and `$MT_CanvasY` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The **pixel** dimensions of the current **device coordinate space** are converted (using an appropriate conversion factor) to **point values** in the **viewport coordinate space**. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to `$MT_CanvasX * 0.5`; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

For more information, see **Table Properties**.

**Scroll Vertically**

The **Scroll Vertically** property becomes available after a value has been set for the **Max Table Height** property. It can take one of two values:

- **Whole table**: The whole table scrolls in and out of the table height that is allotted to the table via the **Max Table Height** 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 **Max 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.

### Wrap Content Auto-Fit Group

You can select different controls, each of which is in a table cell, to belong to a group. Do this via the control's Text Size Auto-Fit property. After you have grouped controls in this way, you can auto-size the text strings in cells of a group so that: (i) all text is the same size, (ii) each text string fits in its respective cell, and (iii) all columns of the group are displayed at an optimum width. In effect, the text size and column widths are optimized so that all of the text in all columns is visible within the available display area. Note that a column will be re-sized only if its width property has been set to wrap_content; if this is not the case, then only the text size will be adjusted while the column remains relatively large.

Note that this optimization is time consuming and should typically be used only on smaller tables.

The property is available on tables. It's values may be any Text Size Auto-Fit group that has been used to identify a set of cells or a template group. The content of the selected group is auto-sized, and the table cells are wrapped within the available width (instead of being cut off). This enables even long rows to be shown within the available width. The property's default value is off.

For more information, see the Text Size Auto-Fit property of any control that features text (such as the Button or Label control).

### Background Color

Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

### Horizontal Alignment

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The
property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property's value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Margin**

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

**Padding**

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.
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 view coordinate space. The view coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the view coordinate space to pixels in the device coordinate space. Mapping the values in this way (from view values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current view (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the view coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Border Width

Sets the border width on all four sides of the table item. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. To set a different border width for any of the four sides, expand the Border Width property to display the individual border width properties (left, right, top, and bottom), and set the different value. For example, if you set Border Width to be 1px and Border Width Top to be 2px, then the widths of the left, right, and bottom borders will all be 1px, while the top border width will be 2px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see `Sizes: Pixels, DPI, DP, SP`.

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 view coordinate space. The view coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the view coordinate space to pixels in the device coordinate space. Mapping the values in this way (from view values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current view (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the view coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.
Border Color
Sets the border color on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border color for any of the four sides, expand the Border Color property to display the individual border color properties (left, right, top and bottom), and set the different value. For example: If you set Border Color to be blue and Border Color Top to be red, then the color of the left, right and bottom borders will all be blue, while the top border color will be red.

Border Style
Sets the border style on all four sides of the table item. Select a value from the dropdown list of the combo box, or double-click in the value field and enter a value. To set a different border style for any of the four sides, expand the Border Style property to display the individual border style properties (left, right, top and bottom), and set the different value. For example: If you set Border Style to be dashed and Border Style Top to be solid, then the style of the left, right and bottom borders will all be dashed, while the top border style will be solid.

Apply Borders to Cells
Select a value of true to apply the table's border properties to the borders of all cells in the table. The properties will be applied only to those table cells that do not have any border property assigned to them. This is a useful property if you wish to apply border properties to all cells at a stroke. If this property is set to true, then changes to a table's border shorthand property will also be passed to the cell borders. Changes to individual table borders (top, right, bottom, or left) will not be passed to cell borders. The default value of the property is false.

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.

Note: Table borders can be conveniently set in the Border Settings dialog (Table | Border Settings).
9.1.21 Time

Time controls are used to format times obtained from a node in a page source. This is useful if you wish to format times in a special way. The formatting is defined in the Date/Time Format String property (see below for details). Note that the source node content must be in the correct lexical format as defined by the XSD specification: `HH:MM:SS`. Optional timezone and millisecond parts are allowed.

Notes

- When the control is associated with a data source node (page source link), placing the mouse over the control (in Page Design View) displays the associated node in a popup.
- All page source links in the page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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.

- All Styles

  The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)
The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map{
  "Bold Text" : $XML1/R/@bold = "1",
  "Italic Text" : true(),
  "Text" : "hello",
  "Text Color" : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size" : $XML1/R/@textsize
}
map{
  "Style Sheet" : "Sheet-1"
}
```

Note the following points:

- When you enter a value for the **All Styles** property—even if it is not a map—all the styling properties of the current component in the **Styles & Properties Pane** will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the **Styles & Properties Pane**.
- You can also specify a style sheet to use, as shown in the second map above.

**Date/Time Format String**

Click the **Additional Dialog** button and enter a date, time, or date-time format in the Format dialog that appears (screenshot below).
The formatting will be applied to the control's content if the content has the correct lexical form of \texttt{xs:date} (for the Date control), \texttt{xs:time} (for the Time control), or \texttt{xs:dateTime} (for the Date, Time, and DateTime controls). Basic examples are:

- \texttt{xs:date}: 2014-12-31
- \texttt{xs:time}: 23:59:59
- \texttt{xs:dateTime}: 2014-12-31T23:59:59

\section*{Control Action}

Click the \textbf{Additional Dialog} button to display the control's \textbf{Actions dialog}. You can set actions to perform when a \textbf{control event} is triggered. The control's event/s are predefined and each is shown in its own tab in the right-hand pane of the \textbf{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 \textbf{Additional Dialog} button. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

\section*{Visible}

An \textbf{XPath expression} that should evaluate to \texttt{true()} or \texttt{false()}. If the expression evaluates to \texttt{false()} — and only if it evaluates to \texttt{false() —} then the control is not visible. If the expression evaluates to \texttt{true()} or returns some other value, then the control is visible. The default is \texttt{true()}. Double-click inside the value field, or click the \textbf{XPath} button, to enter or edit an \textbf{XPath expression}. The \textbf{Visible} property can be used to render an object visible or not depending upon whether an \textbf{XPath expression} evaluates to \texttt{true()}. As a
Controls and Control Events

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. 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'.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.
In MobileTogether Designer, you can use the $MT_CanvasX\textsuperscript{[1]} and $MT_CanvasY\textsuperscript{[2]} dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be $\text{concat}(MT_CanvasX \times 0.5, \ 'px')$.

### Text Size Auto-Fit

Sets whether or not the text size should be automatically reduced to fit the width of the control. You can either select the value from the dropdown list or enter it as an XPath string expression. The property’s values are:

- **off** (the text is not automatically resized; this is the default)
- **ellipsis** (adds an ellipsis at the end of the visible text of the control when that text is too long to be displayed in its entirety)
- **individually** (the text of this control only will be automatically resized)
- **group X** (where $X=1$ to 9). You can set a control to belong to one of nine Auto-Fit groups (where each group is identified by a number from 1 to 9). The text size of all the controls in a group will be automatically resized to that of the control with the smallest of all the auto-fit sizes in that group. This ensures that a selected set of controls has a uniform and reasonable size, so saving you the trouble of finding, by trial and error, the optimum size for a group of controls.
- **template group X** (where $X=1$ to 9). Like for a group (see previous list item), you can set a control in a control template to belong to one of nine Auto-Fit template groups (where each template group is identified by a number from 1 to 9). A template group is different from a group (previous list item) in that, if set on a control template, it is limited to the controls of a control template—as opposed to the controls of a page. However, you can also use a template group as a group if you set it on a page, and not on a control template.

In Design View, text size will be reduced to a minimum size that is 50% of the font size, even if the auto-fit size is smaller. At runtime, however, the actual auto-fit size will be displayed.

**Note**

- If the Multiline property has been set to **true**, then: (i) if auto-fit is disabled, the text will wrap to multiple lines; (ii) if auto-fit is enabled, the text will auto-size and will not wrap; line break characters in the text would, however, create new lines.
- This property cannot be enabled if the Max Number of Lines label property has been set. You must choose either to set a maximum number of lines or to auto-fit the text.
- In tables, this property can be used with the table's Wrap Content Auto-Fit Group property.

### Bold Text

Select **true** or **false** from the dropdown list of the combo box to set the text in bold. You can also use an XPath expression. Default is **false**.

### Italic Text

Select **true** or **false** from the dropdown list of the combo box to set the text in italics. The value can also be generated dynamically via an XPath expression. Default is **false**.
Controls and Control Events

**Underline Text**
Select true or false from the dropdown list of the combo box to set underlining for the text. The value can also be generated dynamically via an XPath expression. Default is false.

**Background Color**
Sets the background color of the object when the object is enabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box; to select the background color of the control, select the item Control's background color in 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. To take up the background color of the control, enter the XPath expression: "control".

**Note:** You can set the background color of the cell to be the same as the background color of the control in the cell, by either: (i) selecting Control's background color in the property's combo box, or (ii) entering the XPath expression "control" in the property's value field.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is disabled, use the Background Color (Disabled) property.

**Background Color (Disabled)**
Sets the background color of the object when the object is disabled. You can do one of the following to select the color:

- Click the color palette to select a background color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Note:** A control can be enabled/disabled according to context. For example, if in a form about personal details, the user enters an affirmative for the holding of life insurance, then the fields for details of the life insurance policy can be enabled; otherwise these fields can be disabled. To enable/disable a field, use the relevant control's Enabled/Editable property. To set a background color for when the object is enabled, use the Background Color property.

**Tooltip**
Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.

**Note:** Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Hint**
Provides a textual hint to the end-user. For example, the hint could be about an action that the end-user has to carry out by using the control. Double-click inside the value field of the property to edit the textual hint, or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Text Color Hint**
Sets the text color of the hint for the control. This text color will be the color of the text that is defined for the Hint property (see above). You can do one of the following to select the color:

- Click the color palette to select a text color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate the required text.

**Horizontal Alignment**
This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**
Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

**Control Width**
Sets the width of the control. Select a value from the property's combo box. The following values are available:

- fill_parent: makes the control as wide as the parent, which could be, for example, a table cell or the page
- wrap_content: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- wrap_content_longest_entry: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, fill_parent creates a maximum width, while wrap_content creates a minimum width. If the combo box is within a table cell, for example, fill_parent would let the combo box fill the cell whereas wrap_content might not fill the cell.

The default value is fill_parent for all controls except the following:

- Image and Chart: For these, the default is wrap_content.
Controls and Control Events

- **Geolocation Map**: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to \( \min(MT_CanvasX, MT_CanvasY) \).

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.

- **Points versus pixels on iOS devices**

  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX$ and $MT_CanvasY$ dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to \( MT_CanvasX * 0.5 \); the XPath expression for this image width would be \( \text{concat}(MT_CanvasX * 0.5, \text{‘px’}) \).

- **Max Control Width**

  This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

  - **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
  - **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

- **Margin**

  Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

  For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see *Sizes: Pixels, DPI, DP, SP*.
Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

### Padding

Sets the control's padding, which is the space between the control's border and the control's content. If the control is a table and the padding is set on a table row or column, then the padding is added to the margin of any control that is contained in the row or column; this increases the inset of the contained control and does not modify the size of table cells.

Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The padding will be set on all four sides of the control. If you wish to set a different padding for any of the four sides, expand the Padding property to display the individual padding properties (left, right, top and bottom), and set the different value. For example: If you set Padding to be 6px and Padding Bottom to be 12px, then the top, left and right padding will be 6px and the bottom padding will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows:...
The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5$; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Tab Order

The Tab Order property takes an integer as its value, or an XPath expression that evaluates to an integer. This integer number is the position of the control in the tab order sequence.

The tab order is the sequence in which controls receive the focus when the client-device user taps the Tab key. The entire tab order sequence can be set quickly and in the visual context of all controls of the page via the **Page | Show/Define Tab Order** 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).

## 9.1.22 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.

▼ All Styles
The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

map{
  "Bold Text" : $XML1/R/@bold = "1",
  "Italic Text" : true(),
  "Text" : "hello",
  "Text Color" : "red",
  "Background Color" : $XML1/R/@background,
  "Text Size" : $XML1/R/@textsize
}

map{
  "Style Sheet" : "Sheet-1"
}

Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, Bold Text and Text Size are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

▼ Line Width
Sets the width (thickness) in pixels of the line. Select a width value (in pixels, dp, or sp) from the dropdown list of the combo box, or double-click in the value field to enter a numeric value. You can also use an XPath expression. The numeric value is understood to be a pixel value. For this reason, no unit
should be specified.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

**Points versus pixels on iOS devices**

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

### Line Style

Specifies the style of the line. You can select one of the options from the dropdown list of the combo box, or use an XPath expression. The default value is solid.

### Line Color

Specifies the color of the line. You can do one of the following to select the color:

- Click the color palette to select a line color
- Select a color from the dropdown list of the combo box
- Double-click in the value field to enter a color code (for example, #FF0000), or click the XPath toolbar button and enter an XPath expression to generate or fetch the required color code

### Visible

An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

**Note:** For information about the visibility of spanned columns/rows, see Table Properties.

**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

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see [Sizes: Pixels, DPI, DP, SP](#).

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values— as numbers — are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be concat($MT_CanvasX * 0.5, 'px').

Style Sheet

The Style Sheet property sets the style sheet to use for the control. 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).

### 9.1.23 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 used 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 segment.
- 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 page source tree are displayed in a bold font. Tree nodes that are not page source links are displayed in normal font.
- Placing the mouse over the page source link in the design tree displays information about the associated control.
- To remove a data-source node association (and therefore the data in the control), right-click the control (in Page Design View) 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 218), select the style or property, and click **Edit XPath** in the pane's toolbar 219.

• **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 220).

• A control's CSS properties can be defined in the Styles & Properties Pane 219 and/or in an external CSS file 220. Those defined in the Styles & Properties Pane 219 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** 220) 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** 220. 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** 220) or paused.

### Video properties

The control's properties are available in the Styles & Properties Pane 219, and are listed below in the order in which they appear.

#### Name

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

#### All Styles

The All Styles property becomes visible if the All Styles setting of the More Project Settings dialog 220 has been set to true. (The default of this setting is false.)

The property enables you to set all of the component's styles via a single XPath map expression, such as the two map expressions below:

```xml
map(
    "Bold Text" : $XML1/R/@bold = "1",
    "Italic Text" : true(),
    "Text" : "hello",
)
```
Note the following points:

- When you enter a value for the All Styles property—even if it is not a map—all the styling properties of the current component in the Styles & Properties Pane will no longer be visible.
- This is an advanced feature, so you must ensure that your XPath map expression is correct, both syntax and values.
- You can enter the key–value pairs of the map in any order.
- The key names are the names of style properties (or styles). In the first map above, for example, **Bold Text** and **Text Size** are style names. The styles that are available for a particular component are listed under that component in the Styles & Properties Pane.
- You can also specify a style sheet to use, as shown in the second map above.

### Video Source

Selects the video source to play in the video control. You can enter a URL that locates a remote file, or enter an XPath expression that selects or generates the URL. In the screenshot below, for example, the URL contained in the last $Sources/AVMedia/Resource element is used to locate the video file to play.

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 page 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
This property is enabled if Image Source Type is url. Sets a user name for user access to the resource. Double-click in the property's value field to edit.

- **Password**
  This property is enabled if Image Source Type is url. Sets a password for user access to the resource. Double-click in the property's value field to edit.

- **Show Controls**
  Specifies whether the buttons of the video control are displayed or not. The values of the property are true or false, with the default being true. If the value is set to false, playback can be controlled via the Video action. 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 event/s are predefined and each is shown in its own 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. Alternatively, you can access a control event by right-clicking the control and selecting the control's event in the context menu that appears.

- **Visible**
  An XPath expression that should evaluate to true() or false(). If the expression evaluates to false()—and only if it evaluates to false()—then the control is not visible. If the expression evaluates to true() or returns some other value, then the control is visible. The default is true(). Double-click inside the value field, or click the XPath button, to enter or edit an XPath expression. The Visible property can be used to render an object visible or not depending upon whether an XPath expression evaluates to true(). As a result, the display of an object can be made to be dynamically dependent on the content or structure of data.

  **Note:** For information about the visibility of spanned columns/rows, see Table Properties.

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

- **Tooltip**
  Sets the text that appears as a tooltip when the end user hovers over the control with the mouse or long presses the control. A tooltip provides useful information to the end user about the control. Double-click inside the value field to edit. If an action has been set for a long press of the control, then no tooltip will be shown on a long press.
Note: Tooltips are not available on all controls, and for some controls they are not available on all platforms. On iOS, tooltips are not available for Edit Field or Signature controls.

**Horizontal Alignment**

This property applies in the case of some controls (such as images and vertical lines) to the control, in other cases (such as radio buttons and check boxes) to the text that accompanies the control. The property sets the horizontal alignment of the control or text to left, center, or right. Default is left for all controls except vertical lines, for which it is center. The property’s value can also be specified via an XPath expression (which enables the value to be generated dynamically).

**Vertical Alignment**

Sets the vertical alignment to top, middle, or bottom. Default is middle. The value can also be specified via an XPath expression (which enables the value to be generated dynamically). For Check Box controls, the property sets the vertical alignment of the check box relative to its text if the text is multiline (see the Multiline property).

**Control Width**

Sets the width of the control. Select a value from the property's combo box. The following values are available:

- **fill_parent**: makes the control as wide as the parent, which could be, for example, a table cell or the page
- **wrap_content**: makes the control only as wide as the control's content requires; when this value is selected, the property Max Control Width becomes available
- **wrap_content_longest_entry**: is available for combo box controls and makes the combo box as wide as the longest content requires; when this property value is selected, the property Max Control Width becomes available
- **percent value**: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- **pixel, dp, or sp value**: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

In effect, fill_parent creates a maximum width, while wrap_content creates a minimum width. If the combo box is within a table cell, for example, fill_parent would let the combo box fill the cell whereas wrap_content might not fill the cell.

The default value is fill_parent for all controls except the following:

- Image and Chart: For these, the default is wrap_content.
- Geolocation Map: The default is the smaller of the two values $MT_CanvasX$ and $MT_CanvasY$. These two dynamic variables give, respectively, the width and height of the device's viewport. Since the default of both Control Height and Control Width are the same (in each case, the smaller of $MT_CanvasX$ and $MT_CanvasY$), the default shape and size of the viewport in the control will always be a square with side equal to $\min(\MT_CanvasX, \MT_CanvasY)$.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.
Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

Max Control Width

This property is available only when the control's Control Width property has been set to wrap_content. The Max Control Width property sets the maximum width of the control. Select a value from the property's combo box. The following values are available:

- percent value: a percentage of the page width; select a value from the dropdown list, or enter a value directly
- pixel, dp, or sp value: select a pixel, dp, or sp value from the dropdown list, or enter a value directly

Initial Width

A value in pixels, dp, or sp that specifies the initial width of the video control. This is the width of the video control while the video is downloading. When the video starts playing, the control's width expands or contracts to the actual width of the video. The default initial width is the value of $MT_CanvasX.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to
these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be `concat($MT_CanvasX \times 0.5, \ 'px')`.

▼ Initial Height

A value in pixels, dp, or sp that specifies the initial height of the video control. This is the height of the video control while the video is downloading. When the video starts playing, the control expands or contracts to the actual height of the video. The default initial height is $(MT_CanvasX / 16) \times 9$.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see `Sizes: Pixels, DPI, DP, SP`.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the `$MT_CanvasX$` and `$MT_CanvasY$` dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX \times 0.5$; the XPath expression for this image width would be `concat($MT_CanvasX \times 0.5, \ 'px')`.

▼ Margin

Sets the margin offsets of the control (or page) relative to the surrounding objects or to the borders of the containing object. Select a value in pixels, dp, or sp from the dropdown list of the combo box, or double-click in the value field to enter a length value. The specified offset will be created on all four sides of the control or page. If you wish to set a different margin for any of the four sides, expand the Margin property to display the individual margin properties (left, right, top and bottom), and set the different value. For example: If you set Margin to be 6px and Margin Bottom to be 12px, then the top, left and right margins will be 6px and the bottom margin will be 12px.

For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see `Sizes: Pixels, DPI, DP, SP`.

Points versus pixels on iOS devices

If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these
values are read as points in the viewport coordinate space. The viewport coordinate space is the canvas on which the design components are drawn, and a point is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the points of the viewport coordinate space to pixels in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components maintain the same size relationship to both the canvas and to each other, no matter what the resolution of the iOS device is or what units are used.

In MobileTogether Designer, you can use the $MT_CanvasX and $MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to $MT_CanvasX * 0.5; the XPath expression for this image width would be `concat($MT_CanvasX * 0.5, 'px')`.

### Style Sheet

The Style Sheet property sets the style sheet to use for the control. 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.

<table>
<thead>
<tr>
<th>Control</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assertion Message</td>
<td>— none —</td>
</tr>
<tr>
<td>Button</td>
<td>OnButtonClicked</td>
</tr>
<tr>
<td>Chart</td>
<td>OnChartClicked</td>
</tr>
<tr>
<td>Check box</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>Combo box</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>Date</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>DateTime (iOS)</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>Edit field</td>
<td>OnTyping</td>
</tr>
<tr>
<td>Geolocation Map</td>
<td>OnGeoMapMarkerClicked</td>
</tr>
<tr>
<td>Horizontal Line</td>
<td>— none —</td>
</tr>
<tr>
<td>Horizontal Slider</td>
<td>OnSliding</td>
</tr>
<tr>
<td>Image</td>
<td>OnImageClicked</td>
</tr>
<tr>
<td>Label</td>
<td>OnLabelClicked</td>
</tr>
<tr>
<td>Radio button</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>Rich Text</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>Signature field</td>
<td>— none —</td>
</tr>
<tr>
<td>Space</td>
<td>— none —</td>
</tr>
<tr>
<td>Switch</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>Table</td>
<td>— none —</td>
</tr>
<tr>
<td>Time</td>
<td>OnFinishEditing</td>
</tr>
<tr>
<td>Vertical Line</td>
<td>— none —</td>
</tr>
<tr>
<td>Video</td>
<td>OnVideoStarted, OnVideoError, OnVideoCompleted</td>
</tr>
</tbody>
</table>
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.
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. 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:

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>What's clicked in the event pane</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl+double-click</td>
<td>Main event (e.g. OnLabelClicked)</td>
<td>Action added as last action of the event</td>
</tr>
<tr>
<td>event/action</td>
<td>Sub-event (e.g. OnClick, OnLongClick)</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Action</td>
<td>Double-clicked action added as next action</td>
</tr>
<tr>
<td>Alt+double-click</td>
<td>Main event (e.g. OnLabelClicked)</td>
<td>Action added as last action of first sub-event</td>
</tr>
<tr>
<td>event/action</td>
<td>Sub-event (e.g. OnClick, OnLongClick)</td>
<td>Action added as last action of sub-event</td>
</tr>
<tr>
<td></td>
<td>Action</td>
<td>None</td>
</tr>
</tbody>
</table>

**Inserting actions from a popup**

The conventional way of inserting an action for processing when an event is triggered is as described above: to drag the action from the Available Actions pane (on the left) to the event pane (on the right). However, you can also insert an action by selecting it from a popup in the event pane.

Insert an action from a popup as follows:

1. Press Alt on your keyboard.
2. Move the mouse cursor to the location where you want to insert the action. The cursor becomes a horizontal line that indicates where the action can be inserted.
3. When the cursor is placed over the insertion location you want, click the mouse. A popup appears that contains an edit field.
4. Enter (i) the first few characters of the name of the action (or action group) you want to insert, or (ii) a text string that is contained in the action's name. The best five matches are shown in the popup.
5. In the popup, select the action (or action group) you want, either (i) by clicking it, or (ii) by using the Up and Down keys to select the action and then pressing Enter. The selected action will be inserted at the insertion location.

**Filtering actions and action groups by name**

To filter actions and action groups by name, enter a part of an action or action group's name in the Quick Filter text box at the top right of the Available Actions pane (see screenshot above). The matched actions and/or action groups will be highlighted in green. Matched action groups will be shown at the top of the Action Groups list, and they will be sorted alphabetically.

The filter feature is useful if you wish to see actions of a particular type. For example, if you wish to display actions related to the saving of data, you could enter Save in the Quick Filter text box.
List usages of an action or action group
Right-click an action or action group in the Available Actions pane, and, in the context menu that appears, click List Usages of this Action. All usages of that action in the design will be displayed in the Listings pane. The usages are listed by page, and then by control.

Order of action execution affects performance
Some actions are executed on the server (generating charts, loading non-embedded files, etc) and some are executed on the client (message boxes, sending an SMS, etc). Therefore, in order to improve performance, the order of actions should be defined in such a way that the switching between server and client for processing is minimized.

Disabling an action
You can temporarily disable any individual action that is defined for an event. A disabled action is ignored, and processing continues as if the disabled action is not defined. To disable an action, right-click it in the definition of the event's actions and select Disable Action. This command is a toggle command, so selecting it once more will enable the action again.

Aborting actions on errors
You can set the project to abort actions when errors occur during action handling. The setting to do this is in the More Project Settings dialog. By default, the Abort option is set to true. The error that triggers the action abortion could be in an XPath expression or elsewhere in the action handling. However, minor errors such as XPath errors for selecting a style property are ignored and action handling continues.

Suppressing logs recursively
Logs are the messages that are displayed in the Messages Pane. 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.

<table>
<thead>
<tr>
<th>Overlay</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Overlay" /></td>
<td>No overlay</td>
</tr>
<tr>
<td><img src="image" alt="Overlay" /></td>
<td>Logs suppressed</td>
</tr>
<tr>
<td><img src="image" alt="Overlay" /></td>
<td>Logs force-enabled</td>
</tr>
</tbody>
</table>

Event pane toolbar
The Event Pane toolbar (screenshot below) offers the following commands, starting from the left:
Actions

- **Undo, Redo**: Undoes and redoes your Event Pane edits
- **Cut, Delete**: Deletes the selected item in the Event Pane. *Cut* additionally puts the selected item on the clipboard
- **Copy**: Copies the selected item to the clipboard
- **Paste, Append Clipboard Contents**: Pastes the clipboard contents relative to the selected location
- **Show/Hide Rarely Used Options**: Some actions have options (or settings) that are inessential to the working of the action. This setting toggles on/off the display of such options
- **Show/Hide Warnings for Actions**: If some mandatory condition required for an action to work correctly is missing, then a warning is displayed in red. This setting toggles on/off the display of warnings
- **Comment Color**: Displays a color picker that lets you set the color of text in the Comment action. The selected color will be applied to all comments in all designs in MobileTogether Designer, including to comments that were defined prior to the setting of the new color
- **Toggle Breakpoint (F9)**: Toggles a breakpoint on/off the selected action
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.

10.1.1 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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be
used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

\texttt{mt-available-languages()}

### 10.1.2 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.

![Send Email To screenshot](image)

**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 From field, the Reply to 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. You can also specify whether the email is sent as HTML or text.

**Server:** If the email is to be sent via the (MobileTogether) server, then MobileTogether Server must be configured to access the ISP's SMTP server. (See the MobileTogether Server documentation for a description of how to do this. Essentially, the ISP's SMTP server address and port, and the sender's email user name and password must be configured in the settings of MobileTogether Server.) When the end user carries out the event that triggers the action, the email is sent silently from the server without any further end-user interaction. Choosing to send the email via the server provides three options more than for sending from the client: the As field, the From field, and the On Error action to perform. The As field specifies whether the email should be sent as HTML or text. The From field is described below. If there is an error when sending the mail from the server, you can specify whether the Send action should be aborted, continued, or throw an error.

---

**To, Cc, Bcc**

The email addresses that go into these fields are entered via XPath expressions. They can be (i) entered directly as strings in the XPath expression (as shown in the screenshots above), or (ii) generated from nodes in page sources (see the XPath expression below). If multiple recipients are to be specified in any of these fields, then it is best to use an XPath expression that returns a sequence. It is not advisable to hard-code separators (such as semi-colons or commas) between two email addresses since different email clients use different separators. Here is an example of an XPath expression that uses a node in an XML page source to generate an email address line:

```
if ( $MT_iOS=true() ) then iosGroup/Person/Email else otherGroup/Person/Email
```

The expression iterates over a sequence of Person/Email nodes, each of which is expected to contain one email address. In the case of both iOS clients and non-iOS clients, multiple recipients are given as a sequence of strings: ("contact1@altova.com", "contact2@altova.com").

---

**From and Reply To**
If you choose to send the email via MobileTogether Server, then the From and Reply To settings become available. You can specify the sender's email address in this setting. The From setting must be filled if the email is sent over an SMTP server that requires a sender's address as mandatory. If the SMTP server does not require this, you can leave the From setting empty.

Automatic replies are often sent from a "pseudo" email address, which is the address made visible to recipients. If you wish to use a "pseudo" address, enter this address in the From setting. If, in this case, you still wish to be contactable, you can enter a real email address in the Reply 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 setting.

Email subject field and body

The XPath expression for the email's Subject field can either be a string or generate the desired text from XML page sources. The XPath expression for the email's Body field must generate structured HTML, that is, an `<html>` element with a valid HTML sub-structure (see below).

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 + icon. 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.

```xml
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

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: `<a href="targetURL">LinkText</a>.

For example:

```
mt-html-anchor('Unregister from mailing list', 'http://www.altova.com')
```

generates the HTML code fragment:

```
<a href="http://www.altova.com">Unregister from mailing list</a>
```

The example above links to an Internet page. For a description of how to link to a MobileTogether solution, see [Hyperlinking to Solutions](#).
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

`mt-email-attachment()`

### 10.1.3 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.

Note: The Share action cannot be used on web clients.

- Message title and text
  
The XPath expressions for these two settings (the message's title and body text) can either be a string or can generate the respective texts from XML page sources.
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 icon. 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(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', $XML/Releases/Release[@date='2015-04-15']/Features, 'XML')` returns the Features node
• **mt-email-attachment**('MTLogo.jpg', $XML4/Images/Image[@name='MTLogo'], 'Base64') returns an image file

The tutorial [Sharing Geolocations](#) shows how the Share action can be used.

**MobileTogether extension functions**

MobileTogether provides a range of **XPath extension functions** that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the **Message Box** action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

**mt-client-ip-address()**

### 10.1.4 Send SMS To

Sends an SMS containing the specified text to the specified telephone number *(see screenshot below).*

Note the following points:

- The recipient's number and the SMS text are specified using XPath expressions.
- The recipient's number must be entered as an XPath string containing numbers. Example expression: 
  
  "004311234567".
- If multiple recipients are planned, the XPath expression must be a sequence of string items, for example: 
  
  
  "(+4311234567", "0011123456789")".
- If the design is generated as an **AppStore App** (see screenshot below), then you can choose whether to send the SMS silently **(Send silently)** or whether the end user sends the SMS, which will be opened in the device's SMS app and await the user's decision **(Use SMS App)**. Note that, for the **Send silently** option to work, (i) the AppStore App will need the **Send SMS** privilege, and (ii) the end user's permission will be asked about whether the app may send messages.
- If the design is deployed as a MobileTogether solution, then the message is opened in the device's SMS app, and the end user is asked whether the message should be sent. This happens even if the **Send silently** option is selected.
- A static global variable named **$MT_SMSAvailable** can be used to test whether SMS services are available on the client device. Values of the variable can be **true()** or **false()**.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

### 10.1.5 Make Call To

Makes a call to the number specified in the XPath expression of the definition (see screenshot below).

- The recipient's number is specified using an XPath expression.
- The recipient's number must be entered as an XPath string containing numbers and no spaces. A plus symbol at the beginning of a number is allowed. Example: `004311234567`.
- If the design is generated as an AppStore App, then the following is possible: (i) the call is started directly (Directly Start Call), or (ii) the number is displayed in the device's telephone app, and the end user is asked whether to dial the number (Use Calling App).
- If the design is deployed as a MobileTogether solution, then, on the action being triggered, the number is displayed in the device's telephone app and the end user is asked whether to dial the number. This happens even if the Directly Start Call option is selected.
- A static global variable named `$MT_TelephonyAvailable` can be used to test whether telephony services are available on the client device. Values of the variable can be `true()` or `false()`.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

### 10.1.6 Open URL/File

Opens, in the client device, the URL or file that is specified in the action. A URL is opened in the client's default Internet browser. A file on the client device is opened in the client's default app for that file type. In the action's configuration options, select the Open URL or Open File in external app radio button and then enter details of the URL or file to open.
In both cases, an option is available for web clients to open the URL in the current tab (or, alternatively, in a new tab of the browser); see screenshots below.

Open URL
In the XPath expression dialog of the Open URL action, if you select Open URL, then you can enter an expression that either: (i) evaluates to a string that is the URL to be opened (first screenshot below), (ii) evaluates to a string that constructs a data url (second screenshot below), or (iii) evaluates to a sequence of strings that constructs a command line instruction (for use in simulations in MobileTogether Designer only).

Data URLs
Data URLs enable you to open binary files directly in a new tab of the web client's browser. Note that these URLs can be accessed in web clients only. The syntax of a data URL is as follows:

```
data: [<mediatype>] [:base64], <data>
```

A data URL consists of four parts (colored differently above). The `data:` part is a fixed prefix. The `<mediatype>` part is a MIME type. The `;base64` token is optional; if used, it indicates that the `<data>` part must be read as Base64-encoded data; if omitted, the `<data>` part will be read as text. The Base64-encoded data can be read dynamically from a node in a page source (for an example, see below).

Example:

```
data: application/pdf;base64, <SomeBinaryData-SuchAsInThe-$BINARIES/PDF-node>
```

In an XPath expression, a data URL that reads Base64-encoded data from a node can be entered as shown in the screenshot below. In this expression, the Base64-encoded data is contained in the $BINARIES/PDF node.

Note: Base64-encoded data in a node can also be saved to file with the Save to Binary File action, and the file can then be opened. The Open Data URL action, on the other hand, opens the file with one click only.
Sequence of strings for a command line instruction

The XPath expression for Open URL can also be constructed so that it evaluates to a command line instruction. In this case, the XPath expression must evaluate to a sequence of strings, where the first string would be the filepath to the executable. The parameters of the command line instruction can then be specified in: (i) a second string of the sequence, where each space-separated token is a command line parameter; (ii) subsequent strings of the sequence, where every subsequent string corresponds to one command line parameter; or (iii) a combination of the previous two patterns (that is, where there is more than one subsequent string, with each of these corresponding to one or more command line parameters).

Note: The parameters would have to be entered in the correct order.

Note: The option for constructing a command line instruction can only be used in a simulation in MobileTogether Designer. It will have no effect when run on a client.

Examples:

The two examples below call Altova DiffDog in order to compare two XML files. The first XPath expression is a sequence of two strings. The second is a sequence of three strings. In both cases, the command line instruction that is constructed will be the same.

```
("C:\Program Files\Altova\DiffDog2021\DiffDog.exe",
 "C:\TestFiles\Colors2.xml  C:\TestFiles\Colors3.xml")
```

```
("C:\Program Files\Altova\DiffDog2021\DiffDog.exe",
 "C:\TestFiles\Colors2.xml",
 "C:\TestFiles\Colors3.xml")
```

Note: If a filepath contains a space, the filepath should be enclosed in quotes (single or double, as required).

Open file in external app

This action opens a file that is located on the client device. You can specify the file directly in the design, or you can let the end-user select the file. The file will be opened in the device's default app for the selected file's filetype. If the client is a web client, the file will be opened in a browser tab; an option is available to select either the current tab or a new tab. Note that, in web clients, file selection by the end-user is not available.

When you click the Open File action's Edit button, an Open dialog appears (screenshot below). Select the options that apply.

File location is stored in design

To directly specify (in the design) the device file to open, select Define file path below (see next screenshot). Enter an absolute or relative file path, or an XPath expression that evaluates to such a file path. If you enter a relative path, then this path is resolved relative to the base directory you specify for that device type (see list and screenshot below).
- **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-back ed-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 (e.g., `txt,xml;html`) or enter an XPath (by pressing the XPath button) that results in a sequence of strings (e.g., `txt/xml/html`).

- **Web Message Box**: Before the File Open dialog is opened on the client device, a message box is displayed. You can enter a message to override the default text of this message box. Enter the text of the message directly, or via an XPath expression.

**Error processing**

The **On Error** option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the **Continue** or **Throw** option.

- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.

- **Throw**: If an error is detected, this option throws an exception that is stored in the **Try/Catch action's variable**. The **Try** 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.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be
used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

`mt-extract-file-extension()`  
`mt-extract-file-name()`

## 10.1.7 Print To

The Print To action (screenshot below) uses Altova StyleVision Server (version 2017sp1 or later) to generate a PDF, Word, or RTF output document. The mechanism works as follows: XML data in one of the design's page source nodes or in an external XML file is processed with a PXF file, which is essentially an XSLT stylesheet. Altova StyleVision Server is used to carry out this XSLT transformation. In order for the Print To action to be carried out, the StyleVision Server application, which is available for download at the Altova website, must be installed on the same computer as MobileTogether Designer (for local simulation) and as MobileTogether Server (for server simulations and deployed use).

### Note about PXF files and StyleVision Server

- A Portable XML Form (PXF) file is a file format that has been specially developed by Altova to package XSLT stylesheets for multiple outputs into one file. The various XSLT stylesheets are generated from a single design created in Altova StyleVision.
- In the design, you can define parameters, which are then also defined in the XSLT stylesheets contained in the PXF file. At runtime, values can be passed to these parameters in the stylesheets.
- StyleVision Server is a lightweight command-line application that is used to run XSLT transformations that generate output in various formats.
- For more information about the PXF format, see the Altova StyleVision documentation. For more information about StyleVision Server, see the Altova StyleVision Server documentation.

The Print To action takes the following options:

**PDF/Word/RTF**

Specifies the type of print output document: `.pdf`, `.docx`, or `.rtf`. You can also use an XPath expression to select the output. The result of the evaluation should be one of the following strings: "PDF", "Word", or
"RTF" (case-insensitive). Using an XPath evaluation enables you to make the selection of print format conditional.

Source
You can select either: (i) an XML node of one of the design's page sources, or (ii) an external XML file, which can be on the client device or on the server.

Source Node
If you choose Node as your source, then enter an XPath expression to select the node you want to use as source data for the output document. The node you select will be entered as the value of this option, and can be changed subsequently.

Source File
If you choose File as your source, the path to the file you select will be entered as the value of the Source File option. You can also add a filepath (or modify an already entered filepath) by clicking the Edit button of the Source File option. A dialog appears in which you can select a server file or client file. See the section File locations below for details of the available options. On selecting a file that is defined to reside on the server, you are asked whether the file should be deployed to the server or not. 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 the solution correctly accesses the source file at runtime.

PXF File
The PXF file is the stylesheet container file that StyleVision Server uses to generate the output document. Select the PXF file using one of the methods described for server locations in the File locations 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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:
• **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the **Continue** or **Throw** option.

• **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.

• **Throw**: If an error is detected, this option throws an exception that is stored in the **Try/Catch action's variable**. 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 Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the *Working Directory* (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the *Working Directory*. See the section **Location of Project Files** for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the **Absolute/Relative Path** field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save**: If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent...
where the action is restricted to file loading.

- **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).

![Screenshot](image)

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 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, Load/Save Binary File, Load/Save Text File, Read Folder, and Get File Info.

**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).

  **Note**: On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the Sessions setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

**Filename is defined by the end user (on the client device)**

- **Default file extension for file saving**: When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter**: The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: txt,html;xml), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: 'txt','html','xml').

- **Optional Default File**: You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- **Web Message Box**: Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- **Automatically create subfolders on file save**: If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note**: On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

**MobileTogether extension functions**

MobileTogether provides a range of [XPath extension functions](https://www.altova.com/mobiledesigner/extension.functions.html) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](https://www.altova.com/mobiledesigner/extension.functions.html).
10.1.8 MapForce Transfer

The MapForce Transfer action enables one set of data structures to be converted (mapped) to a second set of data structures. Each data structure of the output set can be written to a file or to a node of a page source.

To carry out the MapForce transfer, you will need to use the following additional components:

- **Altova MapForce** to design the mapping of one set of input data structures to a set of output data structures. After the mapping has been designed, it is generated as a MapForce Server Execution (MFX or .mfx) file.
- **Altova MapForce Server** must be located on the same machine as MobileTogether Server. It is called by MobileTogether to process the MFX file and generate the output data structures.

**Note:** If you want to test the MapForce Transfer action in a local simulation 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 \texttt{sales-report}, and it will be saved to the root element (\texttt{AllSales}) of the $\texttt{XML2}$ page source. Note that the entire output data structure will be copied to the node specified. So if an XML fragment is generated, then the entire XML fragment is copied to the specified node. You should make sure that the generated data structure will fit correctly into the structure of the target node.

Value Parameters: These are name-value pairs that provide values as inputs for use in the mapping. The name part of a pair is the name of an input component in MapForce; the value part is the value to be passed to this component at run time. You can create as many value parameters as you like.

Note: At run time, the parameter settings you enter for this action are sent as the parameters of a call to MapForce Server. Therefore, each parameter name (across the combined set of input, output, and value parameters) must be unique.

Note: MapForce Server must be installed on the same machine as MobileTogether Server.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, \texttt{mt-available-languages()} returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.1.9  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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

mt-wait-cursor-shown()

10.1.10  Read Contacts

When a Read Contacts action (screenshot below) is added to the design, the $MT_CONTACTS page source tree is automatically added to the design. When the action is triggered, contacts from the device's address book are read and stored in the $MT_CONTACTS tree.
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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.1.11 Access Calendar

When an Access Calendar action (screenshot below) is added to the design, the $MT_CALENDAR page source tree is automatically added to the design. At runtime, 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** must be one of the Boolean values, `true()` or `false()`.

![Screen shot of event creation](#)

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
<?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()">
      <Attendee Name="Bob" Status="Accepted" Type="Required" Relationship="Speaker"/>
    </Event>
    <Event Id="2" Title="New Customer Lunch" Start="2018-05-14T12:30:00" End="2018-05-14T14:00:00" Location="Sushi Restaurant">
      <Attendee Name="Alice" Status="Accepted" Type="Optional" Relationship="Attendee"/>
    </Event>
  </Calendar>
</Root>
```
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.1.12 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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example,
available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

mt-available-languages()

### 10.1.13 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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
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
- Let User Scan Barcode
- Audio
- Audio Recording
- Text to Speech
- Video
- Video Recording
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.

### 10.2.1 Let User Choose Image

The end user can select an image that can be saved to a node of the page source tree (see screenshot below). This enables the end user to select images that are automatically saved to a database. An example of a usage scenario would be the reporting of damages for insurance purposes. The user could run the solution, and take a photograph with the mobile device. The image would be directly uploaded to the appropriate database.

The action has the following properties:

- **Image source**: Select Gallery to let the user choose an image from the image gallery of the client device. Select Camera to start the camera app of the mobile device and capture the next photo taken by the camera.
- **Target Node**: A page source node in which to save the image as Base64-encoded data.

The Let User Choose Image action has three conditions:

- **On OK**: Define actions to perform if the image is correctly imported to target node as Base64-encoded data. Typical actions to perform would be: (i) Reload 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 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 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](#).

**MobileTogether extension functions**

MobileTogether provides a range of [XPath extension functions](#) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the [Message Box](#) action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

```plaintext
mt-base64-to-hexBinary()
mt-hexBinary-to-base64()
mt-hexBinary-to-string()
mt-change-image-colors()
mt-extract-file-extension()
mt-extract-file-name()
mt-image-width-and-height()
mt-string-to-hexBinary()
mt-transform-image()
```

### 10.2.2 Load/Save Image

This action enables the following:

• Loading an image file into a page source node as a Base64-encoded image
• Saving a Base64-encoded image in a page source node as an image file at a server-side or other external location.

**Loading an image file into a page source node**

An image file can be loaded into a page source node by using the **Load Image to Node** option of the Load/Save Image action *(see screenshot below)*. Use an XPath expression to select the target node, that is, the page source node where the image data will be stored. In the **File Path** field, select the image file that is to be loaded into the target node. The image file can be any standard image format (such as BMP, EXIF, GIF, JPG, or PNG). The image file data is converted into Base64 and stored as Base64 encoded data in the target node. Note that the Base64 encoding will containing information specifying the original image format.
Saving Base64-encoded image data as an image file

Base64 image data that is stored in a page source node can be saved as an image file by using the Save Image to File option of the Load/Save Image action (see screenshot below). Select the page source node where the Base64-encoded image is located (the Source Node field; see screenshot below). Then select the location on the server or client where the file is to be saved (the File Path field).

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:

\[
\text{concat('EmployeePhotos/', @name, @surname, '.', suggested-image-file-extension(@photo))}
\]

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 Load Image from File dialog (for loading) or Save Image to File dialog (for saving) appears. In these dialogs, you specify whether the file is located on the server or the client by selecting the respective radio button (see screenshots below).
**File is located on server**

If the image file is located on the server, you can either browse for its location (**Absolute/Relative Path**) or specify the file via a global resource (**File Alias** or **Folder Alias**). Select the option you want.

- **Absolute/Relative Path**: You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the **Working Directory** (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the **Working Directory**. See the section [Location of Project Files](#) for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the **Absolute/Relative Path** field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save**: If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.
- **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).

![Screenshot](image.png)

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 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)*, **Load/Save Binary File** *(Load/Save Text File)*, **Read Folder**, and **Get File Info**.

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)*.

Note: On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the **Sessions** setting in the Misc pane of the Server Settings tab (see the **MobileTogether Server user manual**).

*Filename is defined by the end user (on the client device)*

- **Default file extension for file saving**: When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter**: The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `txt', 'html', 'xml').

- **Optional Default File**: You can enter a default filename, either directly or via an XPath expression, to
guide the end user.

- **Web Message Box**: Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- **Automatically create subfolders on file save**: If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

**Note**: On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

**Error processing**

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.

- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.

- **Throw**: If an error is detected, this option throws an exception that is stored in the *Try/Catch action's variable*. 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.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the *Message Box* action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic *MobileTogether Extension Functions*.

```
mt-last-file path()
mt-base64-to-hexBinary()
mt-hexBinary-to-base64()
mt-hexBinary-to-string()
mt-change-image-colors()
mt-extract-file-extension()
mt-extract-file-name()
mt-image-width-and-height()
mt-string-to-hexBinary()
mt-transform-image()
```
10.2.3 View Image

The View Image action (screenshot below) causes the selected image to be displayed full screen in a new view. The end user can zoom into and out of the image, and can scroll the image both horizontally and vertically.

The image to display can be selected in one of the following ways:

- **Base64 XPath**: The XPath expression must either (i) contain the Base64-encoded text of the image, or (ii) select an XML page-source node that contains the Base64-encoded text.
- **File**: Enter the file path of an image file on the client machine. In the dialog box that appears, select the device directory in which the image file is located, and enter the name of the image file.
- **Control**: Select the control that holds the image. Allowed controls are Image, 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. Choose the control from the dropdown list.

You can also select the following options:

- Whether the image should be displayed at original size or should fill the device screen. The fill-screen option enlarges or reduces the image size so that it fits the screen.
- Whether the image will be rotated according to its EXIF orientation information.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- `mt-base64-to-hexBinary()`
- `mt-hexBinary-to-base64()`
- `mt-hexBinary-to-string()`
- `mt-change-image-colors()`
- `mt-image-width-and-height()`
- `mt-string-to-hexBinary()`
- `mt-transform-image()`
10.2.4 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) are saved, respectively, to the two nodes specified in the definition of the action (Result Node and Result Format Node, respectively). This barcode information is then available to the design as XML data.

The Let User Scan Barcode action (see screenshot above) has the following options:

- **Result Node**: The page source node where the barcode data is saved.
- **Result Format Node**: The page source node where the format of the scanned barcode is saved. The format of the scanned barcode is detected automatically and saved to this node. The format names that are entered in this node are entered exactly as given in the list of supported formats below.
- **Barcode formats for which to scan**: 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.

<table>
<thead>
<tr>
<th>Format name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aztec</td>
<td></td>
</tr>
<tr>
<td>Codabar</td>
<td><strong>Not on iOS</strong></td>
</tr>
<tr>
<td>Code 39</td>
<td></td>
</tr>
<tr>
<td>Code 93</td>
<td></td>
</tr>
<tr>
<td>Code 128</td>
<td></td>
</tr>
</tbody>
</table>
### Data Matrix

<table>
<thead>
<tr>
<th>Code Type</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS: versions &gt;= 8</td>
<td></td>
</tr>
<tr>
<td>EAN-8</td>
<td></td>
</tr>
<tr>
<td>EAN-13</td>
<td></td>
</tr>
<tr>
<td>EAN-128</td>
<td>Detected as Code 128</td>
</tr>
<tr>
<td>ITF</td>
<td>iOS: versions &gt;= 8</td>
</tr>
<tr>
<td>PDF 417</td>
<td></td>
</tr>
<tr>
<td>QR Code</td>
<td></td>
</tr>
<tr>
<td>RSS-14</td>
<td>Not on iOS</td>
</tr>
<tr>
<td>RSS-Expanded</td>
<td>Not on iOS</td>
</tr>
<tr>
<td>UPC-A</td>
<td>iOS: Detected as EAN 13</td>
</tr>
<tr>
<td>UPC-E</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** No barcode support is available in web clients.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

### 10.2.5 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.

![Audio Action Screenshot](image-url)
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 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](#).

![Screenshot of audio settings](image)

- **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-back ed-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

![Screenshot of audio settings](image)
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 page 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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- mt-audio-get-current-position()
- mt-audio-get-duration()
- mt-audio-is-playing()
- mt-audio-is-recording()
- mt-extract-file-extension()
- mt-extract-file-name()

10.2.6 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.
Next, set up the parameters of the recording (see screenshot at the top of the page):

- **Automatic File Extension**: If selected, the file extension will be automatically added depending on the selected format and/or codec. You can use the XPath extension function `mt-extract-file-extension` 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.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- mt-audio-get-current-position()
- mt-audio-get-duration()
- mt-audio-is-playing()
- mt-audio-is-recording()
- mt-last-file path()
- mt-extract-file-extension()
- mt-extract-file-name()

10.2.7 Text to Speech

The Text to Speech action, when set to Start (see screenshot below), converts a text string to speech and plays it back. The text string is either entered directly in an XPath expression or is taken from a node that is selected via an XPath expression. The screenshot below envisions a situation in which, when the main page is loaded, a greeting is played back in the language of the mobile device. The text of the greeting is taken from the Greeting element that has a language attribute with a value that matches the language setting of the mobile device.

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.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.2.8 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 design page are listed by their names in the dropdown list of the Control combo box (see screenshot above). Select the video control to which you want the action to apply; in the screenshot above Video1 has been selected.

Next, select the action you want to execute. The following actions are available:

- **Start**: To play a segment of the video, enter the segment’s From and To times in seconds. If you leave these fields blank, the video will play from start to end.
- **Pause**: Pauses the video when this action is triggered. For example, the Pause action can be defined on a Pause Video button.
- **Resume**: Resumes the video when this action is triggered. It can be triggered, for example, on a button-click event.
- **Stop**: Stops the video when this action is triggered. It can be set, for example, as the action of a Stop Video button.
- **Seek To**: Jumps to the specified position.

**Note:** If a Video control’s 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.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- `mt-video-get-current-position()`  
- `mt-video-get-current-duration()`  
- `mt-video-height()`  
- `mt-video-is-playing()`  
- `mt-video-width()`
10.2.9 Video Recording

The Video Recording action (screenshot below) opens the camera on the mobile device and enables the user to start (and then stop) recording. For example, say the action has been created for an OnButtonClicked event. When the relevant button is clicked in the MobileTogether solution, the client device's video recording app is opened, and the user can start/stop recording. You can specify key properties of the recording, such as: the file name and location, the maximum recording duration, the maximum file size of the recording, and the recording quality. If the video recording action is re-run on the client device, then: (i) if a file name has been specified, the new recording overwrites the previous recording; (ii) if no file name has been specified, the recording is saved as a new file.

Note: Web clients do not support video recording.

You can define the following properties for video recordings:

- **Video File**: On clicking the property's **Additional Dialog** button, the Video Recording dialog (screenshot below) appears.
Here you can specify, either directly or via an XPath expression, the file name or URL of the recorded video file, or skip setting the file name. Note the following points:

- If the file name does not contain a file extension, then the value of the **Optional Default File Extension** setting (in the Video Recording dialog) is appended to the file name. Specify the file extension without a dot (see screenshot above).
- If the file name does not contain a file extension and if no default file extension has been set (previous point), then the recording app's default file extension is appended to the file name.
- If the file name is specified without a file path, then the recording is stored relative to the directory specified in the **Device dependent directories** setting (see screenshot above).
- If no file name is specified, then the recording app saves each recording with a new file name to the device's default video-recording location.
- You can use the MobileTogether XPath extension function `mt-last-file-path` to get the file path of the recording, and the `mt-extract-file-name` function to get the file name from the submitted file path. Note that the `mt-last-file-path` function might not be supported on non-Android devices. Where supported on iOS and Windows devices, it will return the relative path from the base directory that is selected in the **Device dependent directories** setting; on Windows devices this is the Videos directory.

- **Automatic File Extension**: If selected, then the client device selects the file extension on the basis of the recording format or codec. If the **Optional Default File Extension** setting of the previous property has also been set, then this default file extension is used. On Windows devices, however, recordings are always saved in `.mp4` format. (Note that Windows devices will play back non-MP4 formats if these formats are associated with the media player.) The MobileTogether XPath extension function `mt-extract-file-extension` can be used to retrieve the file extension of a file from the the submitted file path. (A file path can be submitted with the `mt-last-file-path` function.)

- **Max. Recording Duration**: The recording will stop once the number of seconds specified in this setting is reached. The number that is entered can be an integer or decimal, and it can be entered as a number (without quotes) or as a string (with quotes). Valid example values are: **10.5**, "**10**".
Note: This setting does not work on some Android devices, and was ignored on the Windows devices that were used in Altova tests.

- **Max. File Size:** The recording will stop once the file size in kB that is specified in this setting is reached. The number that is entered can be an integer or decimal, and it can be entered as a number (without quotes) or as a string (with quotes). Valid example values are: 1000, "5000.5".  
  Note: This setting does not work on Windows and iOS devices.

- **Quality:** Select a value from the combo box. Note that the higher the quality, the greater the file size will be.

**Error processing**

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script:** After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- **Continue:** Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw:** If an error is detected, this option throws an exception that is stored in the *Try/Catch action's variable* (see next). 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.

**MobileTogether extension functions**

MobileTogether provides a range of *XPath extension functions* that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the *Message Box* action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic *MobileTogether Extension Functions*.

- `mt-last-file path()`  
- `mt-extract-file-extension()`  
- `mt-extract-file-name()`  
- `mt-video-get-current-position()`  
- `mt-video-get-current-duration()`  
- `mt-video-height()`  
- `mt-video-is-playing()`  
- `mt-video-width()`
10.3 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.

### 10.3.1 Start/Stop Geo Tracking

The Start/Stop Geo Tracking action (screenshot below) starts and stops tracking the geolocation of the mobile device. After having been started, tracking continues till stopped. Every time when the Read Geo Data 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 information source that is best suited to the solution:

- **GPS+Network**: If the mobile device can be located with GPS, then GPS is used to provide geolocation data. Otherwise, the network provider's geolocator mechanism is used. Geolocation from the network is often less accurate than GPS, but this option has the advantage that geolocation information is always provided. The network thus acts as a backup information source if GPS is not available (for example, inside buildings).

- **GPS**: GPS is used to provide geolocation data. The advantage of using this option is that geolocation data will be accurate. The disadvantage is that, if GPS is not available at a particular location (for example, inside buildings), then no geolocation data will be available.

**Note**: The $MT_GEOLOCATION tree is automatically added to the page sources of the page when the Start/Stop Geo Tracking 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 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).

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

mt-change-image-colors()
mt-geolocation-started()

10.3.2 Read Geo Data
The Read Geo Data action enters the current geolocation data into the $MT_GEOLOCATION tree. In order for the action to be able to read the current geolocation, the mobile device's geolocation tracking must have been started before this action is executed.

The $MT_GEOLOCATION tree is automatically added to the page sources of the page when the Start/Stop Geo Tracking 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=""
</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, E/W)
  - Example: 33°55′11.11″N 22°44′55.25″W

- Degrees, minutes, decimal seconds, with prefixed sign (+/−); the plus sign for (N/E) is optional
  - Example: +33°55′11.11″  −22°44′55.25″

- Degrees, decimal minutes, with suffixed orientation (N/S, E/W)
  - Example: 33°55.55′N 22°44.44′W

- Degrees, decimal minutes, with prefixed sign (+/−); the plus sign for (N/E) is optional
  - Example: +33°55.55′  −22°44.44′

- Decimal degrees, with suffixed orientation (N/S, E/W)
  - Example: 33.33N 22.22W

- Decimal degrees, with prefixed sign (+/−); the plus sign for (N/S E/W) is optional
  - Example: 33.33  −22.22

**Examples of format-combinations:**

33.33N −22°44′55.25″
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.

<table>
<thead>
<tr>
<th>Units and datatypes of retrieved geolocation data</th>
</tr>
</thead>
<tbody>
<tr>
<td>The geolocation data that is retrieved from the different mobile devices is placed in the $MT_GEOLOCATION tree as numbers. The units and datatypes of these numbers are given in the table below.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Android</th>
<th>Web</th>
<th>iOS</th>
<th>Windows Phone</th>
<th>WindowsRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latitude</td>
<td>Degrees (as double)</td>
<td>Decimal degrees (as double)</td>
<td>Degrees (as double)</td>
<td>Degrees (as double)</td>
<td>Degrees (as double)</td>
</tr>
<tr>
<td>Longitude</td>
<td>Degrees (as double)</td>
<td>Decimal degrees (as double)</td>
<td>Degrees (as double)</td>
<td>Degrees (as double)</td>
<td>Degrees (as double)</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Meters (as double)</td>
<td>Meters (as double)</td>
<td>Meters (as double)</td>
<td>Meters (as double)</td>
<td>Meters (as double)</td>
</tr>
<tr>
<td>Altitude</td>
<td>Meters above WGS 84 ref ellipsoid</td>
<td>Meters (as double)</td>
<td>Meters (as double)</td>
<td>Meters (as double)</td>
<td>Meters (as double)</td>
</tr>
<tr>
<td>Speed</td>
<td>Meters/second (as double)</td>
<td>Meters/second (as double)</td>
<td>Meters/second (as double)</td>
<td>Meters/second (as double)</td>
<td>Meters/second (as double)</td>
</tr>
<tr>
<td>Time</td>
<td>UTC time</td>
<td>DOM TimeStamp (unsigned long long)</td>
<td>NSDate (can be converted to TZ)</td>
<td>Int64/System.DateTimeOffset (UTC)</td>
<td>Long long (UTC)</td>
</tr>
</tbody>
</table>
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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

mt-geo-map-marker()
mt-geolocation-started()

10.3.3 Show Geolocation

The Show Geolocation action opens the map application of the mobile device and displays the specified information. You can specify one of the following sub actions (see screenshot below):

- **Show geolocation**: The location on the map of the coordinates that you specify
- **Show route from current location**: The route on the map from the current location to a specified location
- **Show route between locations**: The map route on the map between two specified locations

All locations in these options are given by their geolocation coordinates.

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, E/W)**
  \[ D°M’S.S’S’S’S
  \]
  \[ \text{Example: } 33°55'11.11''N \quad 22°44'55.25''W \]

- **Degrees, minutes, decimal seconds, with prefixed sign (+/-); the plus sign for (N/E) is optional**
  \[ +/-D°M’S.S’S’S’S
  \]
  \[ \text{Example: } 33°55'11.11'' \quad -22°44'55.25'' \]

- **Degrees, decimal minutes, with suffixed orientation (N/S, E/W)**
  \[ D°M.MM’N/S \quad D°M.MM’W/E
  \]
  \[ \text{Example: } 33°55.55’N \quad 22°44.44’W \]

- **Degrees, decimal minutes, with prefixed sign (+/-); the plus sign for (N/E) is optional**
  \[ +/-D°M.MM’’ \quad +/-D°M.MM’’
  \]
  \[ \text{Example: } +33°55.55’ \quad -22°44.44’ \]

- **Decimal degrees, with suffixed orientation (N/S, E/W)**
  \[ D.DDN/S \quad D.DDW/E
  \]
  \[ \text{Example: } 33.33N \quad 22.22W \]

- **Decimal degrees, with prefixed sign (+/-); the plus sign for (N/S E/W) is optional**
  \[ +/-D.DD \quad +/-D.DD
  \]
  \[ \text{Example: } 33.33 \quad -22.22 \]

**Examples of format-combinations:**

- 33.33N \quad -22°44'55.25''
- 33.33 \quad 22°44'55.25''W
- 33.33 \quad 22.45

For information about specifying geolocation data for designer and server simulations, see the section [Geolocation Settings](#).

### MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](#) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the [Message Box](#) action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

- `mt-geo-map-marker()`
- `mt-geolocation-started()`
10.4 NFC

The following actions are available in the NFC group of the Actions dialog (screenshot below):

- NFC Start/Stop
- NFC Push
## Available Actions (use drag&drop):

<table>
<thead>
<tr>
<th>User Interactions</th>
<th>Page</th>
<th>Update Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageBox</td>
<td>Go to Page</td>
<td>Update Node(s)</td>
</tr>
<tr>
<td>Send Email to</td>
<td>Go to Subpage</td>
<td>Insert Node(s)</td>
</tr>
<tr>
<td>Share</td>
<td>Close Subpage</td>
<td>Append Node(s)</td>
</tr>
<tr>
<td>Send SMS to</td>
<td>Scroll To</td>
<td>Delete Node(s)</td>
</tr>
<tr>
<td>Make Call to</td>
<td>Hide Keyboard</td>
<td>Replace Node(s)</td>
</tr>
<tr>
<td>Open URL/File</td>
<td>Update Keyboard</td>
<td>If, Loop, Let, Try/Catch, Throw</td>
</tr>
<tr>
<td>Print To</td>
<td>Restart Display</td>
<td>If-Then</td>
</tr>
<tr>
<td>MapForce Transfer</td>
<td>Update Page Timer</td>
<td>If-Then-Else</td>
</tr>
<tr>
<td>Wait Cursor</td>
<td>Page Sources</td>
<td>Loop</td>
</tr>
<tr>
<td>Read Contacts</td>
<td>Reload</td>
<td>Break Loop</td>
</tr>
<tr>
<td>Access Calendar</td>
<td>Load/Save File</td>
<td>Let</td>
</tr>
<tr>
<td>Let User Choose Date</td>
<td>Load/Save Binary File</td>
<td>Update Variable</td>
</tr>
<tr>
<td>Let User Choose Time</td>
<td>Load/Save Text File</td>
<td>Throw</td>
</tr>
<tr>
<td>Images, Audio, Video</td>
<td>Load/Save HTTP/FTP</td>
<td>Try/Catch Exceptions</td>
</tr>
<tr>
<td>Let User Choose Image</td>
<td>Load/Save String</td>
<td>Try/Catch Server Connection</td>
</tr>
<tr>
<td>Load/Save Image</td>
<td>Load From SOAP</td>
<td>Return</td>
</tr>
<tr>
<td>View image</td>
<td>Save</td>
<td>Action Groups</td>
</tr>
<tr>
<td>Let User Scan Barcode</td>
<td>Delete File/Folder</td>
<td>Manage...</td>
</tr>
<tr>
<td>Audio</td>
<td>Reset</td>
<td></td>
</tr>
<tr>
<td>Audio Recording</td>
<td>Execute SOAP Request</td>
<td></td>
</tr>
<tr>
<td>Text to Speech</td>
<td>Execute REST Request</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>Execute FlowForce Job</td>
<td></td>
</tr>
<tr>
<td>Video Recording</td>
<td>Get File Info</td>
<td></td>
</tr>
<tr>
<td>Geolocation Services</td>
<td>Read Folder</td>
<td></td>
</tr>
<tr>
<td>Start/Stop Geo Tracking</td>
<td>Backup/Restore Page Sources</td>
<td></td>
</tr>
<tr>
<td>Read Geo Data</td>
<td>Show Geolocation</td>
<td></td>
</tr>
<tr>
<td>NFC</td>
<td>Database</td>
<td></td>
</tr>
<tr>
<td>NFC Start/Stop</td>
<td>DB Begin Transaction</td>
<td></td>
</tr>
<tr>
<td>NFC Push</td>
<td>DB Execute</td>
<td></td>
</tr>
<tr>
<td>Push Notifications</td>
<td>DB Bulk Insert Into</td>
<td></td>
</tr>
<tr>
<td>Send Push Notification</td>
<td>DB Commit Transaction</td>
<td></td>
</tr>
<tr>
<td>(Un)Register Ext. PN-Key</td>
<td>DB Rollback Transaction</td>
<td></td>
</tr>
<tr>
<td>(Un)Register PN-Topics</td>
<td>DB Read Structure</td>
<td></td>
</tr>
<tr>
<td>In-App Purchase</td>
<td>Backup/Restore SQLite DB</td>
<td></td>
</tr>
<tr>
<td>Purchase</td>
<td>Miscellaneous</td>
<td></td>
</tr>
<tr>
<td>Restore Purchases</td>
<td>Comment</td>
<td></td>
</tr>
<tr>
<td>Query Purchases</td>
<td>Execute On</td>
<td></td>
</tr>
<tr>
<td>Query Available Products</td>
<td>Execute At Once</td>
<td></td>
</tr>
<tr>
<td>Acknowledge Purchase</td>
<td>Cancel Action Execution</td>
<td></td>
</tr>
<tr>
<td>Get/Report Consumable</td>
<td>User Cancel Behavior</td>
<td></td>
</tr>
<tr>
<td>Get/Report Consumable</td>
<td>Solution Execution</td>
<td></td>
</tr>
<tr>
<td>Get/Report Consumable</td>
<td>Set Language</td>
<td></td>
</tr>
<tr>
<td>Get/Report Consumable</td>
<td>Measure Controls</td>
<td></td>
</tr>
<tr>
<td>Get/Report Consumable</td>
<td>Embedded Message Back</td>
<td></td>
</tr>
<tr>
<td>Get/Report Consumable</td>
<td>Log Message</td>
<td></td>
</tr>
</tbody>
</table>
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.

### 10.4.1 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).

```xml
<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.

**MobileTogether extension functions**

MobileTogether provides a range of [XPath extension functions](#) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the **Message Box** action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

`mt-nfc-started()`

### 10.4.2 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 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.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

`mt-nfc-started()`
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.

10.5.1 Send Push Notification

When the Send Push Notification action (screenshot below) is triggered, a push notification (PN) is sent according to the action's settings (described below). The PN is sent to a receiving solution, which can be: (i) the same solution hosted on the same server, or (ii) another solution that is hosted either on the same server (as the sending solution) or on another server.

The action's settings, which are listed below, define the different parameters of the push notification:
For a MobileTogether Server Service, select Solution or AppStore App

If the Send Push Notification action is being created in a MobileTogether Server Service, then you must specify whether the PN is to be sent to (i) a MobileTogether solution, or (ii) a MobileTogether AppStore App. Select the appropriate radio button to do this. Note that this option is displayed only if the Send Push Notification action is created in a server service; it is not displayed in designs for standard solutions or AppStore Apps.

Server (optional)

If this option is not specified, then the PN is assumed to be targeted at a receiving solution that is being hosted on the same server as the sending solution. If the receiving solution is on a different server than the sending solution, then this server is specified here. The setting's XPath expression must resolve to a string that is the IP address of the relevant server (see screenshot above).

Solution to start (optional)

If this option is not specified, then, on the receiving device, the same solution as the sending solution is started when the PN is received. If the receiving solution is different than the sending solution, then the receiving solution is specified in this setting. The setting's XPath expression must resolve to a string that is the path to the solution's workflow on the server. In the screenshot above, for example, the receiving solution is MyPNReceivingApp, which is located in the public container on the server with the IP address 10.100.10.100.

Send to

Specifies the recipients of the PN. The following options are available:

- **Users**: The PN will be sent to user/s on the receiving-solution's server that are listed in the Users setting. The Users setting takes as its input a single string item (for example: 'User-1') or a sequence of string items (for example: ('User-1', 'User-2')).
- **All users of solution**: All users that have permission to access the receiving solution (on the receiving-solution's server). (User permissions to access a workflow are defined in the administrator settings of MobileTogether Server.)
- **Roles**: The PN will be sent to users that have the role/s specified in the Roles setting. The Roles setting takes as its input a single string item (for example: 'Role-1') or a sequence of string items (for example: ('Role-1', 'Role-2')). (For more information about roles, see the MobileTogether Server documentation.)
- **External PN Keys**: The PN will be sent to devices that have been registered with any one of the external PN keys specified in the External Keys setting. The External Keys setting takes as its input a single string item (for example: 'Key-1') or a sequence of string items (for example: ('Key-1', 'Key-2')). See Register/Unregister Ext PN-Key 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:

```xml
$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.

```xml
$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](https://firebase.google.com/docs/cloud-messaging/http-server-ref).

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

### 10.5.2 (Un)Register Ext PN-Key

This action is used in a solution on a PN-receiving mobile device. It registers a text string as the external PN key of that solution on that mobile device. Note that the device is registered with the specified external PN key for a particular solution. Multiple solutions on a device can each register a different external PN key.
When sending a PN, there are different ways to define a set of PN receivers. One way is to specify that the PN is sent to one or more external PN keys (see Send Push Notification). So, if a device has been registered using one of the targeted external PN keys, then the PN will be sent to that device—more specifically, to the receiving solution on that device that registered the target PN key.

Note: External PN keys can also be used in AppStore Apps.

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 for a solution at any given time, you do not need to specify the key.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.5.3 (Un)Register PN-Topics

This action is used in a solution on a PN-receiving mobile device for registering that device to receive PNs about one or more selected topics. For example, if a device is registered to receive PNs about topics marked as news and topics marked as travel, then it will receive any PN that is marked as either news or travel. A PN is marked with a topic when it is sent (see Send Push Notification). 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')`).

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the *Message Box* action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).
10.6 In-App Purchase

The following actions are available in the In-App Purchase group of the Actions dialog (screenshot below):

- Purchase
- Restore Purchases
- Query Purchases
- Query Available Products
- Acknowledge Purchase
- Get/Report Consumable
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.

10.6.1 Purchase

The Purchase action (screenshot below) sends a purchase request from the client device to the corresponding app store. The purchase request contains the ID of the product being purchased and the user's account ID and profile ID. You can obtain these user IDs from, for example, a user's login credentials or some other suitable database.

When the purchase request reaches the app store, the app store will attempt to carry out the request and bill the user's account according to the user's billing information that is stored at the app store. After the app store executes the purchase request, it sends back information about the purchase. This data will be stored in the $MT_IN_APP_PURCHASE page source, with the buyer's data being stored in the @AccountID and @ProfileID attributes of the most recent Purchase element. You can authenticate the purchase by verifying that the device user's credentials match these two attributes (the buyer's credentials).

Note: If this action is the first In-App Purchase action to be added to the design, then the $MT_IN_APP_PURCHASE page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

mt-client-ip-address()
mt-in-app-purchase-platform-to-product()
mt-in-app-purchase-product-to-platform()
mt-in-app-purchase-service-started()
10.6.2 Restore Purchases

The Restore Purchases action (screenshot below) updates an iOS client with the purchases of the current user account. Purchases made by the current account, including those made from other devices, will be updated on the current device. The action updates the Purchases element of the $MT_IN_APP_PURCHASE page source.

Note that this action applies only to iOS clients; or other clients, use the Query Purchases action.

![Screenshot of the Restore Purchases action](image)

**Note:** If this action is the first In-App Purchase action to be added to the design, then the $MT_IN_APP_PURCHASE page source tree will be added automatically to the design’s page sources. For a description of this page source, see the topic In-App-Purchase Page Source.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- `mt-client-ip-address()`  
- `mt-in-app-purchase-platform-to-product()`  
- `mt-in-app-purchase-product-to-platform()`  
- `mt-in-app-purchase-service-started()`

10.6.3 Query Purchases

The Query Purchases action (screenshot below) queries the respective app store for all purchases made with the current user account, including those made on other devices. The action updates the Purchases element of the $MT_IN_APP_PURCHASE page source.
In the case of iOS clients, a server connection is necessary because the Apple Store only provides a server-side "receipt validation". The receipt validation provides a JSON string that contains the purchase data of the current user of the device. The Purchases element of $MT_IN_APP_PURCHASE is updated with key data of the purchases returned in the JSON string. If you want to use any other data that is available in the JSON string but not added to the Purchases element (for example, the end date of a subscription), you can directly access the JSON string, which is stored in the attribute $MT_IN_APP_PURCHASE/Root/Purchases/@OriginalJSON (see In-App-Purchase Page Source).

Note: For iOS devices that connect to the Apple Store via a server (instead of directly), additional time will be required for the additional steps and processing. So, in these cases, you should consider carefully where in the workflow you are using the Query Purchases action and how this could affect performance.

Note: On iOS devices, you can use the Restore Purchases action to obtain information about the end user's purchases.

Note: If this action is the first In-App Purchase action to be added to the design, then the $MT_IN_APP_PURCHASE page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the MessageBox action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- mt-client-ip-address()
- mt-in-app-purchase-platform-to-product()
- mt-in-app-purchase-product-to-platform()
- mt-in-app-purchase-service-started()

10.6.4 Query Available Products

The Query Available Products action (screenshot below) queries the respective app store for data about the submitted product/s. The submitted products are specified in the Product IDs setting (see screenshot). The value of this setting must be a sequence of strings, each of which is a product name as defined in the In-App...
Purchase Products dialog. If no XPath expression is entered, then all the products defined in the In-App Purchase Products dialog are queried. The data that is returned from the app store is stored in the Products element of the $MT_IN_APP_PURCHASE page source.

**Note:** If this action is the first In-App Purchase action to be added to the design, then the $MT_IN_APP_PURCHASE page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- `mt-client-ip-address()`
- `mt-in-app-purchase-platform-to-product()`
- `mt-in-app-purchase-product-to-platform()`
- `mt-in-app-purchase-service-started()`

**10.6.5 Acknowledge Purchase**

After a purchase request has been made via the Purchase action, the app store attempts to process the purchase transaction. If the payment is successful, then the app store sends the purchase information to the client. It awaits an acknowledgment from the client before before finalizing the payment. If the client does not send an acknowledgment, then it could result in the purchase being canceled. Consequently, it is important to implement the Acknowledge Purchase action so that it occurs soon after the purchase information has been received from the app store. The simplest way to implement the acknowledgment action is as part of the purchasing process.

The Acknowledge Purchase action (screenshot below) sends this acknowledgment. The acknowledgment has to be one of two types:

- *For Non-consumables and subscriptions:* a Purchase acknowledgment.
- *For consumables:* a Consume Purchase acknowledgment.
The Acknowledge Purchase action has two settings:

- Select whether to: (i) Acknowledge Purchase (for non-consumables or subscriptions), or (ii) Consume Purchase (for consumables).
- The purchase token to which the action pertains. The purchase token is a unique value returned by the app store and stored in the $MT_IN_APP_PURCHASE page source tree, in the respective Purchase/@PurchaseToken node.

Note: The Acknowledge Purchase and Consume Purchase actions do not apply to Windows clients.

Note: If this action is the first In-App Purchase action to be added to the design, then the $MT_IN_APP_PURCHASE page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

mt-client-ip-address()
mt-in-app-purchase-platform-to-product()
mt-in-app-purchase-product-to-platform()
mt-in-app-purchase-service-started()

10.6.6 Get/Report Consumable

The Get/Report Consumable action (screenshot below) applies to Windows devices. It enables you to either get (from the Windows app store) your current balance of consumable credit (Get Consumable Balance) or report your consumption of the credit to the Windows app store (Report Consumable Fulfillment).
After a consumable has been purchased, it can be consumed within the app (see Categories of InApp Purchases). In the case of Android and iOS clients, the tracking of consumable credit is handled entirely by the app on the client device. In the case of Windows clients, the Windows app store offers an option to keep a record of a user's current consumable balance. The Get/Report Consumable action addresses this Windows option. It can get the current balance from the Windows app store and report the fulfillment level to the Windows app store.

Fulfillment is different depending on how the consumable is managed:

- **Developer-managed consumable**: The consumable has been fulfilled when all of its credit units have been consumed. The app keeps track of credit consumption and reports fulfillment to the store when all credit units have been consumed. This is necessary to enable the consumable to be purchased again. If fulfillment is not reported, then a new purchase of that consumable will not be possible.
- **Store-managed consumable**: The store keeps track of credit consumption. The consumable does not need to be fully depleted before it can be purchased again. In this case, fulfillment refers to each individual incident of credit consumption. The amount consumed is reported to the store, and the store keeps track of the balance. The balance can be augmented at any time by a new purchase.

You can set up these actions at suitable points in the workflow, depending on the design of the workflow.

The following settings must be entered for the respective actions:

- **Get Consumable Balance**: the Product ID of the consumable that is being queried.
- **Report Consumable Fulfillment**: (i) the Product ID of the consumable that is being reported, (ii) either that the consumable purchase has been fulfilled (in the case of developer-managed consumables) or the amount that has been depleted (store-managed consumable).

**Note**: If this action is the first In-App Purchase action to be added to the design, then the $MT_IN_APP_PURCHASE page source tree will be added automatically to the design's page sources. For a description of this page source, see the topic In-App-Purchase Page Source.

For more information, see Microsoft's Overview of Consumable Add-ons.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
mt-client-ip-address()
mt-in-app-purchase-platform-to-product()
mt-in-app-purchase-product-to-platform()
mt-in-app-purchase-service-started()
# 10.7 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
### Available Actions (use drag&drop):

<table>
<thead>
<tr>
<th>User Interactions</th>
<th>Page</th>
<th>Update Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageBox</td>
<td>Go to Page</td>
<td>Update Node(s)</td>
</tr>
<tr>
<td>Send Email to</td>
<td>Go to Subpage</td>
<td>Insert Node(s)</td>
</tr>
<tr>
<td>Share</td>
<td>Close Subpage</td>
<td>Append Node(s)</td>
</tr>
<tr>
<td>Send SMS to</td>
<td>Scroll To</td>
<td>Delete Node(s)</td>
</tr>
<tr>
<td>Make Call to</td>
<td>Hide Keyboard</td>
<td>Replace Node(s)</td>
</tr>
<tr>
<td>Open URL/File</td>
<td>Update Keyboard</td>
<td>If, Loop, Let, Try/Catch, Throw</td>
</tr>
<tr>
<td>Print To</td>
<td>Restart Display</td>
<td>? If-Then</td>
</tr>
<tr>
<td>MapForce Transfer</td>
<td>MapForce Transfer</td>
<td>? If-Then-Else</td>
</tr>
<tr>
<td>Wait Cursor</td>
<td>Wait Cursor</td>
<td>Loop</td>
</tr>
<tr>
<td>Read Contacts</td>
<td>Read Contacts</td>
<td>Break Loop</td>
</tr>
<tr>
<td>Access Calendar</td>
<td>Access Calendar</td>
<td>Let</td>
</tr>
<tr>
<td>Let User Choose Date</td>
<td>Let User Choose Date</td>
<td>Update Variable</td>
</tr>
<tr>
<td>Let User Choose Time</td>
<td>Let User Choose Time</td>
<td>Throw</td>
</tr>
<tr>
<td>Images, Audio, Video</td>
<td>Images, Audio, Video</td>
<td>Try/Catch Exceptions</td>
</tr>
<tr>
<td>Let User Choose Image</td>
<td>Let User Choose Image</td>
<td>Try/Catch Server Connection</td>
</tr>
<tr>
<td>Load/Save Image</td>
<td>Load/Save File</td>
<td>Return</td>
</tr>
<tr>
<td>View image</td>
<td>Load/Save Binary File</td>
<td>Action Groups</td>
</tr>
<tr>
<td>Let User Scan Barcode</td>
<td>Load/Save Text File</td>
<td>Manage...</td>
</tr>
<tr>
<td>Audio</td>
<td>Load/Save HTML/FTP</td>
<td></td>
</tr>
<tr>
<td>Audio Recording</td>
<td>Load/Save String</td>
<td></td>
</tr>
<tr>
<td>Text to Speech</td>
<td>Load from SOAP</td>
<td></td>
</tr>
<tr>
<td>Video</td>
<td>Save</td>
<td></td>
</tr>
<tr>
<td>Video Recording</td>
<td>Delete File/Folder</td>
<td></td>
</tr>
<tr>
<td>Geolocation Services</td>
<td>Execute SOAP Request</td>
<td></td>
</tr>
<tr>
<td>Start/Stop Geo Tracking</td>
<td>Execute REST Request</td>
<td></td>
</tr>
<tr>
<td>Read Geo Data</td>
<td>Execute FlowForce Job</td>
<td></td>
</tr>
<tr>
<td>Show Geolocation</td>
<td>Get File Info</td>
<td></td>
</tr>
<tr>
<td>NFC</td>
<td>Read Folder</td>
<td></td>
</tr>
<tr>
<td>NFC Start/Stop</td>
<td>Backup/Restore Page Sources</td>
<td></td>
</tr>
<tr>
<td>NFC Push</td>
<td>Database</td>
<td></td>
</tr>
<tr>
<td>Push Notifications</td>
<td>DB Begin Transaction</td>
<td></td>
</tr>
<tr>
<td>Send Push Notification</td>
<td>DB Execute</td>
<td></td>
</tr>
<tr>
<td>(Un)Register Ext. PN-Key</td>
<td>DB Bulk Insert Into</td>
<td></td>
</tr>
<tr>
<td>(Un)Register PN-Topics</td>
<td>DB Commit Transaction</td>
<td></td>
</tr>
<tr>
<td>In-App Purchase</td>
<td>DB Rollback Transaction</td>
<td></td>
</tr>
<tr>
<td>Purchase</td>
<td>DB Read Structure</td>
<td></td>
</tr>
<tr>
<td>Restore Purchases</td>
<td>Backup/Restore SQLite DB</td>
<td></td>
</tr>
<tr>
<td>Query Purchases</td>
<td>Miscellaneous</td>
<td></td>
</tr>
<tr>
<td>Query Available Products</td>
<td>Comment</td>
<td></td>
</tr>
<tr>
<td>Acknowledge Purchase</td>
<td>Execute On</td>
<td></td>
</tr>
<tr>
<td>Get/Report Consumable</td>
<td>Execute At Once</td>
<td></td>
</tr>
<tr>
<td>Cancel Action Execution</td>
<td>User Cancel Behavior</td>
<td></td>
</tr>
<tr>
<td>Solution Execution</td>
<td>Set Language</td>
<td></td>
</tr>
<tr>
<td>Measure Controls</td>
<td>Embedded Message Back</td>
<td></td>
</tr>
<tr>
<td>Log Message</td>
<td>Return</td>
<td></td>
</tr>
</tbody>
</table>

© 2015-2021 Altova GmbH
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.

### 10.7.1 Go to Page

When triggered, the GoToPage action goes to the specified page. The page to go to is selected in the combo box of the action (see screenshot below) or via an XPath expression. If no other page exists in the project, no page will be available for selection in the combo box. In the screenshot below, the GoToPage action is placed below the two sub-events On Click and On Long Click. This defines that the action is triggered when the button is either tapped (clicked briefly) or pressed (clicked for a longer time).

The Load Page Message option (see screenshot above) enables a message to be shown in a progress dialog while the page loads. If you want to display a message, check this option, and then define the message as an XPath expression. If the option is unchecked, no message will be displayed.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

### 10.7.2 Go to Subpage

Goes to the specified sub-page when triggered. The subpage to go to is selected from the Go to Subpage combo box (see screenshot below), which lists all the subpages in the project. In the screenshot below, an OnLabelClicked event has been set to go to the Orders subpage of the MobileTogether project. Alternatively, the name of the subpage can be obtained via an XPath expression, for example: "Orders" or $XML3/DataSources/Subpage[@id="Name"]/@name.
The Go to Subpage action has the following settings:

- **Parameters**: If parameters have been declared on the selected sub page, then each parameter of that sub page is now listed below the sub page entry, and a value can be passed to each parameter. In the screenshot above, for example, a value is defined for the parameter named \( a \). It is an error if no value is passed to a mandatory parameter. If the sub page is not selected via the combo box but by using an XPath expression, then the name of the sub page is not generated till runtime. Since no parameters can, therefore, be known at design time, instead of any parameter names being displayed, an entry for **Parameters** is displayed. Here, you can enter an XPath map expression or array expression to define parameter values, for example: map\{"a": "a-value", "second-param": "second-param-value"\} or \[("a-value"), ("second-param-value")\]. If you enter an array expression, (i) the order of sequences in the array must match the order of parameter declarations of the sub page, and (ii) values for optional parameters may not be left out of the array (an empty sequence can be entered for a parameter value that you wish to leave empty). Note that you can use other parameters when defining the value of a parameter.

- **Source XML**: Specifies whether and how subpage data should correspond to page sources in the originating page. This definition is entered in the Subpage Data Mapping dialog, which is accessed by clicking the **Additional Dialog** button of the **Source XML** field. See the section "Source XML of subpages" 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 **OnBackButtonClick** 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 \textit{Width} and \textit{Height} settings.

If the optional \textit{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 \textit{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 \textit{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 \textit{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.

\textbf{Source XML of subpages}

Every subpage must have at least one page source in order for the subpage to be used effectively. Data in page sources of subpages can be mapped to data in top page sources, or the data in subpage sources can be left unmapped. If mapped, then subpage data is transferred to the mapped nodes of the top page when the subpage is closed. The behavior of the source XML of subpages is defined in the Subpage Data Mapping dialog (screenshot below left). To access this dialog, click the \textbf{Additional Dialog} button of the \textit{Source XML} field of the \textit{Go to Subpage} action definition (see description above).

The following subpage data handling possibilities are available:

- Subpage data is not mapped. In this case, page sources of subpages are handled independently of the subpage-data-mapping mechanism.
- A subpage source is mapped to a top page structure that is exactly the same; even the names of elements and attributes are the same.
- A subpage source is mapped to a top page source that has a different structure and/or different attribute/element names.
Subpage Data Mapping

- Do not map data to subpage
- Map data from source element to subpage root:
  Choose Source XML Element as well as the Root for the Subpage
  `$WORKFLOWDATA1.PE` ➔ `$SUBPAGEDATA1`

Choose where the data should be mapped to (optional)

PrivateAddresses
- Sources
  - Address
  - ZIP
  - City
  - Street

Addresses
- Sources
  - Address
  - ZIP
  - City
  - Street

Drag and drop the elements/attributes from the left pane to the right pane to create mappings between the data. The "bold" items in the right pane must be mapped as they are used on the Subpage. Items with grey text color cannot be dragged/dropped.

Choose XML

```
- Married
- HasAddresses
- LastName
- FirstName
- Birthday
- Birthtime
- Comment
- Phone
- Email
- Income
- Picture
- PrivateAddresses
  - Address
    - ZIP
    - City
    - Street
- CompanyAddresses
```

OK Cancel
In the Subpage Data Mapping dialog (screenshot above left), the page source tree of the subpage is displayed in the right-hand source tree pane, while the page source tree of the top page is displayed in the left-hand source tree pane.

- Select the subpage page source from the combo box above the (right-hand) subpage pane. This combo box lists all the page sources of the subpage. Note that the subpage name is listed above the subpage pane.
- To select the top-page page source to be mapped, first click the Additional Dialog button above the (left-hand) top page pane. This displays the Choose XML dialog (screenshot above right), which contains all the page sources of the top page. Select the node that you want to map to the subpage data. In the screenshot above, the PrivateAddresses node has been selected in the Choose XML dialog. As a result, the descendants of PrivateAddresses are displayed in the top page source tree pane and PrivateAddresses, the mapped node, is listed above the pane.

In the case described by the screenshot above, the mapping is automatically done between the two page sources since both XML data structures are identical, right down to the names of elements. In the screenshot below a mapping between elements is constructed by dragging a node from the left-hand pane onto a node in the right-hand pane. This is because the two data structures are not identical (the first element, $, 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).
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.7.3 Close Subpage

Closes the active subpage when triggered. Optionally, when the subpage is closed, a return value can be saved. The return value is generated by an XPath expression that you specify (see screenshot below). The value can be retrieved as the Subpage Result of the Let action.

```
$onButtonClick: 'CalculateAndClose'

$onClick: [on Enter, on Escape]

$onCloseSubpage: [on Close Subpage and return (optionally)]

$onLongClick: [on Long Click]
```

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.7.4 Scroll To

You can specify that, when the action is triggered, the solution scrolls to one of the following:

- Top or bottom of the active page (screenshot below).
- Top or bottom of the selected table. The table is selected by choosing its name from the combo box or by entering an XPath expression that evaluates to the table’s name. Only tables that have vertical scrolling are listed for selection in the Table option’s combo box.
- A row-group of a table that is identified by name. The table’s name is selected from a combo box or entered as an XPath expression. You can optionally specify an XPath condition to select a specific row group (of the selected table). The first row group for which the condition evaluates to true will be the target of the Scroll To action.
- A control that you select by its name (see screenshot below). Select the name from the combo box or create an XPath expression that evaluates to the name of the control. If you select a control in a repeating-table structure, then you can specify an additional condition, via an XPath expression, to refine the search for the control. The dynamic variables that return control information, such as $MT_ControlValue, can be used to locate a control in a dynamically changing context.

**Desired location in the view**
The Desired Location combo box becomes available when the Row Group or Control option is selected. It enables you to specify where in the view the target object must be shown. A value of Minimum Scroll brings the target into view with the minimum of scrolling. So, for example, if you scroll to an Image control, minimum scroll will just bring the full image into view. A value of Top scrolls the target so that it is at the top of the view. Center scrolls the target to the center of the view, and Bottom scrolls it to the bottom of the view.

**MobileTogether extension functions**
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.7.5  Hide Keyboard

Hides the keyboard of the mobile device when the action is triggered. Some mobile devices automatically show a keyboard when certain page elements, such as text fields, are present or active on the page. The Hide Keyboard action is useful if you wish to gain some screen space for the page display on such devices.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.7.6  Update Display

Updates all the controls of a page with the data currently contained in each control's corresponding page source node. This enables the display explicitly to be updated immediately after an action is executed. For example, if the Update Display action is used inside a loop action, then the display is updated when the actions in each loop iteration are executed.

Updating selected controls/nodes

By default, all the controls of a page are updated with data held in the corresponding page source nodes. You can, however, update only a selection of controls. The selection is done by specifying, via XPath, the page source nodes that provide the data of these controls.

To update only a selection of nodes, do the following:

1. Go to the More Project Settings dialog (by double-clicking the value field of the More Project Settings property in the Project section of the Styles & Properties Pane).
2. Set Advanced UpdateDisplay Options to true and close the dialog.
3. If you now add an Update Display action to any event, the action will provide the four options listed below. Select the option you want:
   - **All** updates all controls on the page (with data currently contained in the corresponding page source nodes). This is equivalent to the default setting when advanced options are disabled (see above).
   - **All Except** updates all controls on the page except those that are linked to the nodes selected by the option's XPath expression.
   - **Only** updates only those controls that are linked to the nodes selected by the option's XPath expression.
   - **None** updates no control and the display remains unchanged.

**Note:** When the Update Display action is executed in a control or page context, only the variables currently available to the control or page, respectively, can be executed. Variables inside an Action Group will not be available outside the Action Group, and so cannot be used in the XPath expressions to select controls/node to update (see list above). If you want to use a variable in these XPath expressions, define the variable at a global level or set its value as the content of a node in a page source (for example, in the $PERSTENT page source).

### MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](#) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

### 10.7.7 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.

### MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](#) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be
used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.8 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 Text File**
- **Load/Save HTTP/FTP**
- **Load/Save String**
- **Load from SOAP**
- **Save**
- **Delete File/Folder**
- **Reset**
- **Execute SOAP Request**
- **Execute REST Request**
- **Execute FlowForce Job**
- **Get File Info**
- **Read Folder**
- **Backup/Restore Page Sources**
The actions in this group are available for page events and control events. The fastest way to access the Actions dialog (screen shot above) is to right-click the page or control and select the page/control actions command. See also Page Events and Control Events.

### 10.8.1 Reload

Reloads the external resource specified in the action's combo box (see screenshot below). External resources referenced by the project are available as entries in the combo box, and include XML files, charts, and images.

![Reload screenshot](image_url)

Note the following:

- If the resource to be updated is a page source, then an XPath expression that locates the root node of the page source can be used (for example: $XML1 or $DB2).
- Notice that the screenshot above contains entries for chart controls. If selected for update, the chart image will be updated when the event is triggered. Image controls can also have their linked images updated with the Update 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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- `mt-external-error-code()`
- `mt-get-source-from-name()`
- `mt-get-source-name()`
- `mt-get-source-structure()`

10.8.2 Load/Save File

You can set the action to either: (i) load data from a file, or (ii) save data to a file. To specify whether it is a load action or a save action that is carried out, select the appropriate radio button (see screenshots below).

Load from file

For each LoadFromFile action, you can select one page source from the action's combo box (see screenshot below). Alternatively, you can use an XPath expression that locates the root node of the page source (for example: `$XML1`). You can then specify a file from which to load data for that page source when the event is triggered.

![LoadFromFile screenshot]

To load data for multiple page sources when the event is triggered, add multiple LoadFromFile actions, as shown in the screenshot above.

Error processing

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue
or Throw option.

- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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 selected page source or page source node (both are selected via the first XPath expression), to the XML file specified in the File Path field (see screenshot below). The encoding of the XML file is specified in the Encoding field. To save data for multiple data sources, add multiple SaveToFile actions.

Note that you have the option of selecting either the entire page source (Use Page Source) or a subtree of a page source (Use Source Node).

**Error processing**

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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 Save File dialog appears. In this dialog, you specify whether the file is located on the server or the client by selecting the respective radio button (see screenshots below).
**File is located on server**

If the file is located on the server, you can either browse for its location (*Absolute/Relative Path*) or specify the file via a global resource (*File Alias* or *Folder Alias*). Select the option you want.

- **Absolute/Relative Path:** You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the **Reset** button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the **Working Directory** (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the **Working Directory**. See the section **Location of Project Files** for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the *Absolute/Relative Path* field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.
- **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).

![Folder Alias Screenshot](image)

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 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**, **Load/Save File**, **Load/Save Image**, **Load/Save Binary File**, **Load/Save Text File**, **Read Folder**, and **Get File Info**.

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).

  **Note**: On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the Sessions setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

**Filename is defined by the end user (on the client device)**

- **Default file extension for file saving**: When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter**: The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: txt, html; xml), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: 'txt', 'html', 'xml').

- **Optional Default File**: You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- **Web Message Box**: Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- **Automatically create subfolders on file save**: If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note**: On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

**Error processing**

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
• **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.

• **Throw**: If an error is detected, this option throws an exception that is stored in the **Try/Catch action's variable**. 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.

### MobileTogether extension functions

MobileTogether provides a range of **XPath extension functions** that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the **Message Box** action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic **MobileTogether Extension Functions**.

- `mt-last-file path()`
- `mt-external-error-code()`
- `mt-extract-file-extension()`
- `mt-extract-file-name()`
- `mt-external-error-code()`
- `mt-get-source-from-name()`
- `mt-get-source-name()`
- `mt-get-source-structure()`
- `mt-save-json-to-string()`

### 10.8.3 Load/Save Binary File

The Load/Save Binary File action enables the following:

- Loading a binary file into a page source node as Base64-encoded content
- Saving the Base64-encoded content of a page source node as a binary file to the client or to a server-side location.

This allows binary files, such as PDFs, to be held as XML content.

#### Loading a binary file into a page source node

A binary file can be loaded into a page source node by using the **Load Binary to Node** option of the Load/Save Binary File action (see screenshot below). Use an XPath expression to select the target node, that is, the page source node where the binary data will be stored. In the **File Path** field, select the binary file that is to be loaded into the target node. The binary file data is converted to Base64 and stored as Base64-encoded content in the target node.
Saving Base64-encoded content as a binary file

Base64-encoded content that is stored in a page source node can be saved as a binary file by using the Save Binary to File option of the Load/Save Binary File action (see screenshot below). Select the page source node where the Base64-encoded content is located (the Source Node field; see screenshot below). Then select the location on the server or client where the file is to be saved (the File Path field).

Binary file locations

When you click the Additional Dialog button of the File Path field of the Load/Save Image action (see screenshots above), the Save Binary to File dialog appears. In this dialog, you specify whether the file is located on the server or the client by selecting the respective radio button (see screenshots below).

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 Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the Working Directory (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the Working Directory. See the section Location of Project Files for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the Absolute/Relative Path field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

**Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

**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 Global Resource Definitions for details.
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).

![Screenshot showing global resource folder alias]

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 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, Load/Save Binary File, Load/Save Text File, Read Folder, and Get File Info.

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 standing 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).

  **Note**: On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the Sessions setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

**Filename is defined by the end user (on the client device)**

- **Default file extension for file saving**: When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Optional File Filter**: The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: `txt,html;xml`), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: `'txt'`, `'html'`, `'xml'`).

- **Optional Default File**: You can enter a default filename, either directly or via an XPath expression, to guide the end user.

- **Web Message Box**: Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

- **Automatically create subfolders on file save**: If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

  **Note**: On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

**Error processing**

The **On Error** option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the **Continue** or **Throw** option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.

- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- `mt-last-file path()`
- `mt-external-error-code()`
- `mt-external-error-text()`
- `mt-external-error-code()`
- `mt-get-source-from-name()`
- `mt-get-source-name()`
- `mt-get-source-structure()`

### 10.8.4 Load/Save Text File

You can set the action to either: (i) load data from a text file to a page source node, or (ii) save data from a page source node to a text file. To specify whether it is a load action or a save action that is carried out, select the appropriate radio button (see screenshots below).

**Load from text file**

For each LoadTextFile action, you can select one target node from a page source. For example, in the screenshot below, the Description node is the target node. Next, specify a text file from which to load text data into the target node. You can also select a default encoding. If the encoding of the text cannot be found automatically, then the default encoding will be used.

To load data for multiple target nodes when the event is triggered, add multiple LoadTextFile actions.
Error processing
The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the **Continue** or **Throw** option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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 text file
Saves data from the selected page source node to the text file specified in the File Path field (see screenshot below). The encoding and end-of-line characters of the text file are specified, respectively, in the Encoding and End of Line fields.

To save data for multiple page sources, add multiple SaveNode actions.

Error processing
The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the **Continue** or **Throw** option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action is used to specify what action to take if an error occurs.
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 Text File action (see screenshots above), the Save Text to File dialog appears. In this dialog, you specify whether the file is located on the server or the client by selecting the respective radio button (see screenshots below).

File is located on server
If the file is located on the server, you can either browse for its location (Absolute/Relative Path) or specify the file via a global resource (File Alias or Folder Alias). Select the option you want.

- **Absolute/Relative Path:** You can enter a path, browse for a file, or enter an XPath expression that generates the path to the file. Use the Reset button to remove the current entry. The path can be relative to the design file, or absolute. If the file is deployed to the server along with the design file, then the relative/absolute path specified in the dialog will be used internally (in the server's database) to access the file. If the file is not deployed, then it must be stored in a directory on the server. In this case: (i) if a relative path is selected in the Load From or Save/Specify File dialog, then, at runtime, this relative path will be resolved on the server with reference to the Working Directory (defined in the MobileTogether Server settings); (ii) if the path in the Load From or Save/Specify File dialog is absolute, the file's containing folder on the server must be a descendant of the Working Directory. See
the section Location of Project Files for details. You can also choose whether to allow untrusted SSL connections or not, when accessing or saving the file. If the Absolute/Relative Path field is in a dialog to save a file—and not to load a file—you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

- **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the server, they will be created when the file is saved. This option is relevant only when saving; it is absent where the action is restricted to file loading.

- **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).

![Screenshot](image.png)

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 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, Load/Save Binary File, Load/Save Text File, Read Folder, and Get File Info.

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).

  **Note:** On web clients, files are stored temporarily on the server. They are deleted from there when the server session ends. A server session ends after a specified period of inactivity; this period is defined in the Sessions setting in the Misc pane of the Server Settings tab (see the MobileTogether Server user manual).

  **Filename is defined by the end user (on the client device)**

  - **Default file extension for file saving:** When saving files, you can optionally specify a default file extension; this extension will be used if none is specified with the file name.

  - **Optional File Filter:** The browse dialog that is opened on the client device will filter the file types to be loaded/saved so that only those file extensions that you have defined are allowed. You can enter: (i) a comma-separated or semicolon-separated list of extensions (for example: txt, html; xml), or (ii) an XPath expression that returns a sequence of string items, where each string item is a file type extension (for example, here is one sequence containing three string items: 'txt', 'html', 'xml').

  - **Optional Default File:** You can enter a default filename, either directly or via an XPath expression, to guide the end user.

  - **Web Message Box:** Before the File Open/Save dialog is opened, a message box is displayed. You can enter text directly or via an XPath expression to override the default text of the message box.

  - **Automatically create subfolders on file save:** If intermediate folders in the file path are missing on the client, they will be created when the file is saved. This option is relevant only when saving; it is absent if the action is a file loading action.

    **Note:** On iOS devices, letting the user select the file on the device works only as an import/export from/to the iCloud; users are not allowed to browse the backed-up folder or non-backed-up folder.

**Error processing**

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script:** After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
• **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.

• **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.

**MobileTogether extension functions**

MobileTogether provides a range of **XPath extension functions** that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic **MobileTogether Extension Functions**.

```
mt-last-file path()
mtparameter-

```

10.8.5 **Load/Save HTTP/FTP**

You can set the action to either: (i) load data from an HTTP/FTP file, or (ii) save data to a file via HTTP/FTP. To specify whether it is a load action or a save action that is carried out, select the appropriate radio button (see screenshot below).

**Load from HTTP/FTP**

For each **LoadFromHTTP/FTP** action, you can select one page source from the available page sources and specify an HTTP/FTP source from which to load data. (Alternatively, you can use an XPath expression that locates the root node of the page source (for example: `$XML1`).) When the event is triggered, data from the HTTP/FTP source will be loaded into the page source you have specified. To load data for multiple page sources, add multiple **LoadFromHTTP/FTP** actions.

![OnPageLoad](image)
Error processing
The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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 page source that is selected in the action's combo box to an XML or HTML file at a target HTTP or FTP location that is specified in the Settings field of the action's definition (see screenshot below). (Alternatively, you can use an XPath expression that locates the root node of the page source (for example: $XML1).) To enter access details of the HTTP/FTP location, click . This displays the Edit Web Access Settings dialog 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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action is used to specify what action to take if an error occurs.
occurs. If no error occurs, then the next action is processed. See the section Try/Catch action for details.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.8.6 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 XPath expression of the Source Node setting (see screenshot below), and generates the structure and data of the selected page source (see screenshot). Select the page source from the Page Source combo box, or an XPath expression that locates the root node of the page source (for example: `$XML1`).

The type of the page source selected (JSON or XML) should correspond to the serialization of the string. So, if the source node string is a JSON string, then a JSON document will be generated. If the source node string is an XML string, then this string is loaded into the page source, which must be an XML page source; the root element in the serialized string will be created as the root element of the XML page source.

- **About the source string:** In the screenshot above, an XML string is located in the source node `$SERIALIZEDSTRINGS/Strings/XML_String`. This is the string that will be loaded into the selected page source. In order for this to work correctly, the page source must be an XML page source.
XPath expression of the Source Node setting need not take a string from a node; the string can also be entered directly in the XPath expression.)

- **Encoding of the source string:** The following encodings are supported on clients, so the source string must be in one of these encodings: UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32BE, US-ASCII, ISO 8859-1.

- **About the JSON/XML page source:** The page source needs to have been created before it is selected in the action. The JSON/XML document that is loaded from the string is created as the entire page source within the document node. If, at run time, the structure of the loaded document does not match the structure of the page source as defined in the design, then the solution will not be executed correctly. This is because the design works with the node names of an expected page source structure, but the nodes of the page source that is actually created have different names.

**Save to String**

The Save to String action ([screenshot below](#)) serializes the page source named in the Page Source option, and saves the serialized string to the location that is specified by the XPath expression of the Target Node setting. You can also select the page source via the setting's combo box or via an XPath expression that locates the root node of the page source (for example: `$XML1`).

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\MobileTogetherDesigner7\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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

```
mt-load-json-from-string()
mt-load-string()
mt-save-json-to-string()
mt-string-to-hexBinary()
```
10.8.7 Load from SOAP

For each `LoadFromSOAP` action, you can load data from a SOAP request that is generated from a WSDL file. To choose the WSDL file, click the **Additional Settings** button of the *File Path* field (see screenshot below). The data is loaded into the selected page source.

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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the `Continue` or `Throw` option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (`On Success`), or when there is an error (`On Error`). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the `Try/Catch action's variable`. 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.

**MobileTogether extension functions**

MobileTogether provides a range of [XPath extension functions](#) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be
used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.8.8 Save

Saves data of the page source that is selected in the action's combo box to the default file of that page source. The data source must be an editable XML file or DB. To save data to multiple data sources, add multiple Save actions. Note that page sources that are read as JSON will also be saved as JSON (not as XML even though the data is presented in the GUI as an XML tree); also see Page Source Options.

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 page source. You can specify which columns can be updated or can take inserted values. (Updates refer to modified data in existing row elements; inserted values refer to data in newly added row elements.) By default, the Insert and Update options of each column are selected together as a pair. If, however, you wish to specify different options for a column's Insert and Update options, check the Use separate filter settings for Insert and Update statements check box. Attributes with empty values can be converted to NULL values in the DB by checking that column's NULL check box. Note that missing attributes will always be saved as NULL.

Columns that cannot be updated (because they are user-defined, fixed-value, or calculated-value) will not have an Insert, Update, or NULL option check box. In the screenshot above, the ID column cannot be updated because it holds fixed values. Deselect the columns you do not want to update.

You can specify the order in which deletions, updates, and insertions occur by selecting the desired order in the combo box at the bottom of the dialog.

If you wish to reset the Save settings so that all columns are updated, click Reset to default.

**Error processing**

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script:** After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
Continue: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.

Throw: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- mt-external-error-code()
- mt-external-error-text()
- mt-get-source-from-name()
- mt-get-source-name()
- mt-get-source-structure()

### 10.8.9 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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is
the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.

- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.

- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, \texttt{mt-available-languages()} returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.8.10 Reset

Resets the page source that is selected in the action's combo box to the default values defined in the Page Sources Pane. The page source can also be entered via an XPath expression (for example: \texttt{$XML1$}). Note that you can reset any kind of page source that has been defined in the Page Sources Pane, including the \texttt{$PERSISTENT$}, \texttt{$MT\_GEOLOCATION$}, \texttt{$MT\_FILEINFO$}, and \texttt{$MT\_NFC$} page source trees.

\begin{itemize}
  \item \textbf{Note}: If a variable locates a root node (\texttt{$XML$, $PERSISTENT$, etc}) and its infoset is changed because the Reset action is executed, then the variable is automatically updated to locate data in the new infoset. Note that this applies only when infoset modifications are caused by the Reset action (as opposed to, say, a data update caused during the execution of some other action).
\end{itemize}

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, \texttt{mt-available-languages()} returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
# 10.8.11 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** and **Execute FlowForce Job** actions. 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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the **Continue** or **Throw** option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.8.12 Execute REST Request

Executes a REST request that you define in the RESTful API Request dialog. To open this dialog, click the Additional Settings button of the Settings field (see screenshot below).

![Screenshot of Additional Settings button](image)

After you have defined the REST request, the URL of the request is displayed in the Settings 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 and Execute FlowForce Job actions. So the variable will contain the last result generated by any of the actions that use it.

If you wish to change the REST request after one has already been defined, click the Additional Dialog button of the Settings field (see screenshot above). This displays the RESTful API Request dialog, in which you can define the new request.

Error processing

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.
details.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.8.13 Execute FlowForce Job

Executes a FlowForce job request that you define in the Edit FlowForce Settings dialog. To open this dialog, click the Additional Settings button of the Settings field (see screenshot below).

![Screenshot](image)

After you have defined the FlowForce job request, the URL of the job is displayed in the Settings field of the action. The FlowForce job will be executed at runtime, and the result will be returned. If you wish to store the result of the request in the $MT_HTTPExecute_Result 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 and the Execute REST Request actions. So the variable will contain the last result generated by any of the actions that use it.

If you wish to change the FlowForce job request after one has already been defined, click the Additional Dialog button of the Settings field (see screenshot above). This displays the Edit FlowForce Settings dialog, in which you can define the new request.

Error processing

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.8.14 Get File Info

The Get File Info action (screenshot below) adds information about the specified file (or folder) to a page source called `$MT_FILEINFO`. An `$MT_FILEINFO` page source is created either when a Read Folder action or a Get File Info action is added to the design, whichever is added first. In the settings of the action, select the file about which you want to get info.

Note that there is only one `$MT_FILEINFO` page source per page. So, if multiple Get File Info actions are triggered during the processing of a page at run time, then `$MT_FILEINFO` at any given time will contain information from the last Get File Info action that was triggered.

Structure of the `$MT_FILEINFO` tree

The structure of the `$MT_FILEINFO` tree is as shown in the screenshot below. The `Root` element has a number of attributes that will be filled with the file information of the file specified in the triggered action. Descriptions of the attributes are given below.

- **Path**: The full path of the file being reported.
- **Size**: The file size in bytes.
- **Creation Time**: 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 Write Time.
- **Access Time**: The time when the file was last accessed.
- **Write Time**: The time when the file was last written to.
- **IsDirectory**: Can take a value of true or false.
- **IsReadOnly**: Can take a value of true or false.

**Note**: The data read by the Get File Info action is passed to the attributes of $MT_FILEINFO/Root (as described above). The $MT_FILEINFO/Root node, however, also has child elements: repeating File elements, which receive data via another action, Read Folder (see screenshot). The Read Folder action fills the File elements with metadata about files in a specified folder.

**Example**

An example file named ReadFolderGetFileInfo.mtd shows how to use the Get Info action (see screenshot of simulation below). This file is available in the (My) Documents folder Altova\MobileTogetherDesigner7\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.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

**mt-cache-update-dateTime()**
When a Read Folder action (see screenshot below) is added to the design, the $MT_FILEINFO page source, which is structured as a set of repeating File elements, is created in the design. In the Read Folder action, you specify the folder that you want to read (see screenshot and example below). At run time, the contents of the specified folder (files and sub-folders) are read, and metadata information of each file and subfolder of the target folder is placed in a corresponding File element of the $MT_FILEINFO page source.

Note the following points:

- The folder can be specified either by browsing for it or by entering an XPath expression that evaluates to a string that is the path to the target folder.
- If the option to Recurse into subfolders is checked, then files and subfolders in recursive subfolders are read. Otherwise, only items in the specified folder are read.
- If the Recurse into subfolders option is checked, then the option to include empty directories becomes available. This enables information about empty folders to be included in the data that is read out.
- The File Pattern setting uses wildcards to filter which files of a folder are read. For example: ‘*.mp3’ reads the details of all .mp3 files in the folder; ‘My*.*’ reads the details of all files that have names that begin with the characters My and take any suffix; ‘*’ or ‘’ reads the entire folder. If recursion into subfolders has been enabled, then the pattern is applied recursively to subfolders.
- To enter multiple file patterns, the XPath expression must be a sequence of string items, each of which specifies a single pattern. For example: ("*.xml", "*.txt").
- While the Read Folder action provides information about files and subfolders of the target folder, it does not provide information about the target folder itself. For details of the folder itself, submit the name of the folder in the Get File Info action.
- Each page contains one $MT_FILEINFO page source. So, if there are multiple Read Folder actions on a page, then, at run time, $MT_FILEINFO at any given time will contain information about the folder that was read by the last Read Action to have been triggered.
- The one $MT_FILEINFO page source is also populated with data obtained by the Get File Info action. While the data read by the Read Folder action is passed to the File child elements of $MT_FILEINFO/Root, data read by the the Get File Info action is passed to the attributes of $MT_FILEINFO/Root.
The $MT_FILEINFO page source is created either when a Read Folder action or a Get File Info action is added, whichever is added first.

Example
An example file named ReadFolderGetFileInfo.mtd shows how to use the Read Folder action (see screenshot of simulation below). This file is available in the (My) Documents folder Altova\MobileTogetherDesigner7\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.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.8.16 Backup/Restore Page Sources

Each Backup/Restore Page Sources action consists of three sub-actions (see screenshot below), of which one may be selected.

The three sub-actions are:

- **Backup Page Sources**: Backs up an internal copy of one or more page sources. Select the page source/s to backup by clicking the Additional Dialog button at the bottom of the action and selecting one or more of the available page sources. (see screenshot above).
- **Restore Backed-up Page Sources**: After an internal copy of a page source has been backed up (with Backup Page Sources), a page source might be further modified. The Restore Backed-up Page Sources action returns the page source to the state of the most recent internally saved copy.
- **Discard Backed-up Page Sources**: After an internal copy of a page source has been backed up (with Backup Page Sources) and that page source has been further modified, the Discard Backed-up Page Sources action discards the previously backed-up internal copy.

**Note**: The Backup/Restore Page Sources action applies only to page sources that have been backed up temporarily—that is, they are not saved to file. To save to file, use other actions like Save or Save File.

**Usage**
The Backup/Restore Page Sources action enables you to back up a page source temporarily, and then to accept or discard further modifications on the basis of whether one or more conditions are met. For example, before going to a subpage, a page source can be backed up internally. After the user has edited data on the sub page, they can press buttons, respectively, to confirm or cancel the changes. The buttons would be defined, respectively, to discard and restore backed up page sources.

**MobileTogether extension functions**
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.9 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**
- **DB Read Structure**
- **Backup/Restore SQLite DB**

**Note:** These actions are used to interact with data in DB page sources. They are not suitable for displaying data. If you wish to display data from a DB page source, then insert (into the design) a control that is linked to a page source node. For information, see the sections about controls and page 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.

10.9.1 DB Begin Transaction

When the event is triggered the DB Begin Transaction action begins a transaction with the data source selected in the Connection combo box. This combo box lists all the data sources of the project, and also offers the option of setting up an additional database connection specifically for use with the DB Begin Transaction action.

The Transaction Locking option specifies the level of protection, and enables you to ensure that data is not corrupted during write actions. The following options are available:

- **Database Default**: Collects DB-related default settings on the DB, server, and client.
- **Prevent from writing of affected tables**: The DB will not be written to if it is currently being written to via another connection. The Write-transaction will be deferred till the other Write-transaction has been completed; otherwise an error message will be displayed.
- **Prevent from reading and writing of affected tables**: The DB will not be read from or written to if it is currently being written to via another connection. The transaction will be deferred till the other transaction has been completed, or an error message will be displayed.

If you connect to an SQLite database, a *Timeout (in seconds)* property becomes available. This enables you to specify a wait period for applying a write-lock. If no timeout period is specified, SQLite will abort immediately if a write-lock is not applied on transaction begin.

**Error processing**

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.

- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.

### MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

```
mtparam-available-db-connection-names()  
mtparam-db-any-changed-fields()  
mtparam-db-any-changed-rows()  
mtparam-db-deleted-original-fields()  
mtparam-db-deleted-original-rows()  
mtparam-db-file-path()  
mtparam-db-modified-fields()  
mtparam-db-modified-rows()  
mtparam-db-new-fields()  
mtparam-db-new-rows()  
mtparam-db-original_row()  
mtparam-external-error-code()  
mtparam-external-error-text()  
```
10.9.2 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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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 page sources. It is not the ideal mechanism for displaying data. If you wish to display data from a DB page source node, then insert (into the design) a control that is linked to a page source node. For information, see the sections about controls and page 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 page source. In the first line below, a fixed value is used; in the second line, the parameter Maker is used.

WHERE Manufacturer= 'BMW'
WHERE Manufacturer= :Maker

To use a parameter, write the parameter name prefixed by a colon (:) in the SQL statement where you want to use it. As soon as you enter the first character after the colon, an entry is created for the parameter in the Parameters pane. Next, in the Parameters pane, enter an XPath expression to provide the value of the parameter. You can enter as many parameters as you like.

Note: In the SQL statement, column and table names from the source database are used, since the SQL statement directly queries the DB. In the XPath expression of parameters, however, you must use the names of nodes in page source trees (Row, RowSet, etc), since it is these trees in which values related to the design are stored.

Statement built with XPath
You can also use XPath to build an SQL statement. Select Statement built with XPath, and enter the XPath expression that generates the required SQL statement. The advantage of this is that it provides greater flexibility in creating the SQL statement. For example, you can include design tree nodes, other XPath constructs, and end user input to calculate and generate parts of the SQL statement.
To build an SQL statement using XPath, select *Statement built with XPath*. In the *Edit XPath/XQuery Expression dialog* 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:

```xml
<DB>
  <RowSet>
    <Row Model="23 COUPE 2014"/>
    <Row Model="23 COUPE 2015"/>
    <Row Model="23 COUPE 2016"/>
    <Row Model="24 3.0 SI COUPE 2014"/>
    <Row Model="24 COUPE 2015"/>
  </RowSet>
</DB>
```

- To access the model name of the first car in the returned nodeset, the XPath expression would be: `$MT_DBExecute_Result/DB/RowSet/Row[1]/@Model`.
- To access the row of the car with the year 2016 in its model name, the XPath expression would be: `$MT_DBExecute_Result/DB/RowSet/Row[@Model[contains(. , '2016')]]`.

**MobileTogether extension functions**

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the **Message Box** action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

- `mt-available-db-connection-names()`
- `mt-db-any-changed-fields()`  
- `mt-db-any-changed-rows()`  
- `mt-db-deleted-original-fields()`  
- `mt-db-deleted-original-rows()`  
- `mt-db-file-path()`  
- `mt-db-modified-fields()`  
- `mt-db-modified-rows()`  
- `mt-db-new-fields()`  
- `mt-db-new-rows()`  
- `mt-db-original row()`  
- `mt-external-error-code()`  
- `mt-external-error-text()`

### 10.9.3 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. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

  - **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the **Continue**
or **Throw** option.

- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (**On Success**), or when there is an error (**On Error**). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the **Try/Catch action's variable**. 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.

**MobileTogether extension functions**

MobileTogether provides a range of **XPath extension functions** that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, **mt-available-languages()** returns the languages in which the solution is available and could, for example, be used with the **Message Box** action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic **MobileTogether Extension Functions**.

```
mt-available-db-connection-names()
mt-db-any-changed-fields()
mt-db-any-changed-rows()
mt-db-deleted-original-fields()
mt-db-deleted-original-rows()
mt-db-file-path()
mt-db-modified-fields()
mt-db-modified-rows()
mt-db-new-fields()
mt-db-new-rows()
mt-db-original-row()
mt-external-error-code()
mt-external-error-text()
```

### 10.9.4 DB Commit Transaction

When the event is triggered the DB Commit Transaction action commits a transaction to the page source selected in the **Connection combo box**. This combo box lists all the page sources of the project, and also offers the option of setting up an additional database connection specifically for use with the DB Commit Transaction action.

![Diagram](image)

**Error processing**

The **On Error** option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—
and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.

- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.

- **Throw**: If an error is detected, this option throws an exception that is stored in the *Try/Catch action's variable*. 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.

### MobileTogether extension functions

MobileTogether provides a range of *XPath extension functions* that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, *mt-available-languages()* returns the languages in which the solution is available and could, for example, be used with the *Message Box* action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic *MobileTogether Extension Functions*.

- mt-available-db-connection-names()
- mt-db-any-changed-fields()
- mt-db-any-changed-rows()
- mt-db-deleted-original-fields()
- mt-db-deleted-original-rows()
- mt-db-file-path()
- mt-db-modified-fields()
- mt-db-modified-rows()
mt-db-new-fields()
mt-db-new-rows()
mt-db-original-row()
mt-external-error-code()
mt-external-error-text()

10.9.5  DB Rollback Transaction

When the event is triggered the DB Rollback Transaction action rolls back a transaction on the page source selected in the Connection combo box. This combo box lists all the page sources of the project, and also offers the option of setting up an additional database connection.

Error processing
The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

mt-available-db-connection-names()
mt-db-any-changed-fields()
mt-db-any-changed-rows()
mt-db-deleted-original-fields()
mt-db-deleted-original-rows()
mt-db-file-path()
mt-db-modified-fields()
mt-db-modified-rows()
mt-db-new-fields()
mt-db-new-rows()
mt-db-original row()
mt-external-error-code()
mt-external-error-text()

10.9.6 DB Read Structure

The DB Read Structure action enables data to be read from a specified database and stored in a page source named $MT_DBSTRUCTURE. This page source is filled exclusively by data that is obtained when the DB Read Structure action is executed.

Defining the DB Read Structure action

When the DB Read Structure action is dropped in the Event pane, a $MT_DBSTRUCTURE page source is added to the design (visible in the Page Sources Pane). The DB that will be read is defined in the action’s settings (screenshot below). These settings are described below.
Read structure from
Specifies the location of the DB structure; solution or server. The structure can be one of the DB page sources in the solution, or it can be a DB that is accessed via a connection stored on MobileTogether Server. For information about stored DB connections, see the description of server-side DB connections in the MobileTogether Server user manual.

Note: Server-side DB connections are available only on Windows-based MobileTogether Servers. As a result, on Linux-based or macOS-based MobileTogether Servers, you are restricted to reading DB structures that are contained in the solution.

Connection name
The connection name can be entered as an XPath string value (see screenshot below). If the structure's location has been specified to be the solution (see previous point), then the connection name is also available for selection in a combo box.

Read Actual/Possible Structure:
The $MT_DBSTRUCTURE page source has a structure that is a superset containing components that are available across various types of DBs.

- **Read Actual Structure**: Reads the structure of the given DB connection. You can select components to read, as well as filter these components by name (see screenshot above).
- **Read Possible Structure**: That subset of nodes of the $MT_DBSTRUCTURE page source will be filled that matches the structure of the given DB connection and for which data can be returned. Tables of the DB structure read in this way are not identified by their names.

Filter
This option is displayed only if the Read Actual Structure option (see previous option) is selected. It enables you to filter what DB components to read. You can filter the components in one of the following ways:

- **By Selection**: Select the check boxes of the components to be read (see screenshot above). To further filter the selected component by name, specify an XPath expression which is a sequence of strings that gives the names of the component to be read. If a particular component has ancestor-type components, then the ancestor-type components will automatically also be read. For example, if a column has been selected, then the column's table ancestor will automatically also be read.

- **By XPath**: The XPath expression must be a sequence of arrays (see screenshot below). The first item in each array is the component type to read (tables, columns, etc); these items are known as keywords, and available keywords are displayed in a popup that appears when you hover over the option's XPath button; keywords are case-insensitive. The subsequent items of the array (second item onwards) are the names of that component type to read. For example, in the XPath expression shown in the screenshot below, the columns named **Author** and **Publisher** of tables named **Book** are read.

![OnPageLoad for page 'New Page1'](image)

**Error processing**

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.

- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.

- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. 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.

**What happens at runtime**

At runtime, the specified DB is read, and the nodes of the $MT_DBSTRUCTURE page source are filled with data from the DB. The data in this page source can now be used in the design.

**Note**: A MobileTogether XPath extension function named `mt-available-db-connection-names` can be used to get the names of all available DB connections, either from the solution or the server.
Simulations

If you use the server for simulation, make sure that the server settings in MobileTogether Designer are correctly set and that the DB is available in the server side solution's working directory. See the section Simulation on Server for more information.

If you run a simulation directly in MobileTogether Designer, the data that will be used in the simulation will come from the DB that is specified in the Read DB-Structure Simulation setting (which is available in the Simulation 2 tab of the Options dialog).

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

- mt-available-db-connection-names()
- mt-db-any-changed-fields()
- mt-db-any-changed-rows()
- mt-db-deleted-original-fields()
- mt-db-deleted-original-rows()
- mt-db-file-path()
- mt-db-modified-fields()
- mt-db-modified-rows()
- mt-db-new-fields()
- mt-db-new-rows()
- mt-db-original-row()
- mt-external-error-code()
- mt-external-error-text()

10.9.7 Backup/Restore SQLite DB

This action (screenshot below) enables you to backup an SQLite database multiple times to a folder you designate. You can subsequently restore the SQLite database from one of these backups. This feature is available only on MobileTogether Server Advanced Edition.
To set backups, do the following:

1. Select *Backup*.
2. Select the SQLite DB connection that you want to back up.
3. Select the folder where the backups should be stored. This can be a path that is relative to the server-side solution's working directory or an absolute path. The filename of the backed up SQLite database will be generated automatically and is a concatenation of (i) the DB connection name that you entered in Step 2 (for example, *Contract_Management* in the screenshot above) and (ii) the current timestamp (in the format `YYYY-MM-DD HH-MM-SS`). So, for example, a possible filename would be: *Contract_Management 2021-06-18_10-30-24*, where the numbers are the date and time.
4. Optionally, enter the maximum number of backups. After this number is exceeded, the oldest backup will be deleted. If no value is set or if a value of 0 is set, then an unlimited number of backups is allowed.

To restore the SQLite database from a backup, do the following:

1. Select *Restore*.
2. Select the connection to the SQLite DB that you want to restore.
3. Select the relative or absolute path of the backup file from which to restore. Relative paths must be relative to the server-side solution's working directory.

**Note:**
- After a backup has been created, the path to it can be obtained by calling the `mt-last-file-path()` function. Note that this function returns the full file path.
- When a SQLite database is restored via the Restore action, any new data in the replaced database file (since the last backup) will be lost. If you want to keep this data make sure that you back up the database before restoring it.
- After a backed up file is restored via the Restore action, the backup file is not deleted. After a Restore action, the `mt-last-file-path()` function returns the full path of the backup file (that was used for the restore).
- If the DB is locked when a restore is attempted, an error is returned. There are no retries or timeouts.

**Error processing**

The *On Error* option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script:** After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the *Continue* or *Throw* option.
- **Continue:** Actions are not terminated. Instead, you can select what to do in either event: when there is no error (*On Success*), or when there is an error (*On Error*). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw:** If an error is detected, this option throws an exception that is stored in the *Try/Catch action's variable*. 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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

```
mt-available-db-connection-names()
mt-external-error-code()
mt-external-error-text()
```
10.10  Miscellaneous

The following actions are available in the Miscellaneous group of the Actions dialog (*screenshot below*):

- Comment
- Execute On
- Execute at Once
- Cancel Action Execution
- User Cancel Behavior
- Solution Execution
- Set Language
- Measure Controls
- Embedded Message Back
- Log Message
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.

10.10.1 Comment

Comments can be added to the definition of an event's actions (see screenshot below). This is useful for inserting explanations of the different actions in the definition of the event's actions.

You can customize the color of comment text. To do this, click the Comment Color icon in the event pane's toolbar and select a color from the color picker that appears. The selected color will be applied to all comments in all designs opened in MobileTogether Designer, including to comments that were defined prior to the setting of the new color.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

`mt-available-languages()`

10.10.2 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.

**MobileTogether extension functions**
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the **Message Box** action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).
10.10.3 Execute at Once

If a variable contains a nodeset from a page source and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. There are two situations in which alternative processing is desirable:

- Changes to a nodeset occur via a sequence of multiple actions, each of which changes the nodeset. This leads to a situation in which each action is processing a nodeset that has been modified by a previous action.
- The evaluation of an XPath expression containing a variable that references a modified page source (for example, if the XPath expression occurs within a Loop action).

The Execute at Once action does the following:

1. Evaluates the XPath/XQuery statements of all its child actions
2. Applies the effects of these evaluations to the relevant nodesets of the page.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.10.4 Cancel Action Execution

The Cancel Action Execution action (highlighted in the screenshot below) cancels the execution of the event's action sequence. In the screenshot below, for example, if an exception is thrown in Try part of the Try/Catch action, 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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.10.5 User Cancel Behavior

The User Cancel Behavior action (highlighted in the screenshot below) enables you to override a user cancellation action and continue with action execution. The following options are available:

- **Cancel all actions**: If the user presses the **Back** button (or the **Cancel** button that appears during the execution of long actions), the cancellation is allowed. This is the default behavior.
- **Request cancellation**: This option enables you to override a user cancellation action. See the example in the screenshot below. If the user attempts to cancel action execution, the action execution is not interrupted. Instead, the mt-user-tried-to-cancel-actions 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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.10.6 Solution Execution

When the event is triggered, the Solution Execution action provides you with the option to: (i) cancel the solution, (ii) leave it suspended (that is, running in the background), or (iii) restart the solution (with the same parameters as on the solution start).

If you select either of the first two options (cancel or suspend), you can do one of the following: (i) switch to another solution, (ii) switch to the Solutions Overview (the Solutions page of MobileTogether Client), (iii) do not switch to either another solution or the Solutions Overview. The various possibilities are given in the table below.

<table>
<thead>
<tr>
<th>Switch to...</th>
<th>Cancel solution</th>
<th>Suspend solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Another solution</td>
<td>Solution is closed without displaying any message, and the specified solution is opened.</td>
<td>The current solution runs &quot;minimized&quot; in the background, and the specified solution is opened.</td>
</tr>
<tr>
<td>Solution Overview</td>
<td>Solution is closed without displaying any message, and the display switches to the Solutions Overview page of MobileTogether Client.</td>
<td>The current solution runs &quot;minimized&quot; in the background, and the display switches to the Solutions Overview page of MobileTogether Client.&quot;Switch to&quot;</td>
</tr>
<tr>
<td>&quot;Switch to&quot;</td>
<td>Solution is closed without displaying any message, and the display switches to</td>
<td>The current solution runs &quot;minimized&quot; in the background, and the display switches to where it</td>
</tr>
</tbody>
</table>
Note the following points:

- Web clients do not support suspended solutions. However, the current solution can be kept running while another solution is opened in another browser tab and becomes the active solution.
- The *Open in new browser tab* option is available for web clients only. It starts the solution in a new browser tab; the original solution is not stopped.
- A solution that is suspended (running in the background) is displayed minimized as an icon in the *Running* page of the MobileTogether Client application, and can be opened by tapping this icon.
- If the solution runs “minimized” in the background, then the solution is paused at that point, and no further solution action is executed. For example, no timers are executed, no geolocations are used, and audio playback is paused. When the solution is re-opened, actions defined for the *On Reopen* option of the *OnPageRefresh* 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.
- For Web clients only: When the option to switch to a new solution is selected, the *Token and Audience* settings can be entered as XPath string expressions. These two entries are required only for setting up solutions for authenticated users; they can be left empty if you do not want to silently authenticate users when they run the second solution.
- If the *Switch to* option is unchecked, the display either: (i) returns to the home screen if the solution was started via its shortcut, or (ii) returns to the *Solutions Overview*. On iOS, it always returns to the *Solutions Overview* page.
- Tapping the *Back* button on the top page of a solution running in parallel suspends the solution. (i) *On Android, Windows App and Windows*: It returns to the home screen if the solution was started via its shortcut; otherwise it returns to the *Solutions Overview*. In either case, the return action is performed directly, without any end-user input. (ii) *On iOS (and in non-parallel solutions)*: The user is asked whether the solution should be exited or not. If the response is to exit, then the user is returned to the *Solutions Overview* page.

**MobileTogether extension functions**

MobileTogether provides a range of *XPath extension functions* that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the *Message Box* action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic *MobileTogether Extension Functions*.

### 10.10.7 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 the language setting of the mobile device must correspond to the code name of one of the solution's localization languages.

However, you can also let the user select the language of the solution without needing to change the language of the device. The Set Language action enables the solution to be restarted with a user-selected language. For example, the user can tap a button to change to a specific language if this action is triggered by a button click (see screenshot below, which shows an action to restart the solution in US Spanish). Another option is to let the user select a language from a dropdown list in a combo box.

Enter the language-country code (for example, es-US or fr-CH) or just the language code (for example, es or fr). If the action is triggered (to select a specific language), then the solution's language will be decided according to the cascading order given below.

1. If the solution contains a matching language-country (es-US or fr-CH) localization, then strings of this localization are used where these exist
2. If no matching language-country localization (es-US or fr-CH) exists for a string, then the localized language (es or fr) string is used—if one exists
3. If no language-country localization (es-US or fr-CH) or language localization (es or fr) exists for a string, then the default language of the solution is used for that string

Alternatively, use the default language of the mobile device. In this case, the language setting of the device determines what language from among the available localization languages will be used. The same set of cascading rules listed above applies.

If you wish to see a simulation in any of the languages for which localized strings are defined, set the simulation language via the Project | Simulation Language command, and then run a simulation.

**Error processing**

The On Error option lets you define what should be done if an error occurs. Since the error handling can be precisely defined for this action, errors on such actions (that provide error handling) are treated as warnings—and not errors. The advantage is that you do not need to check errors on actions for which error handling has already been defined. The following error handling options are available:

- **Abort Script**: After an error occurs, all subsequent actions of the triggered event are terminated. This is the default action if an error occurs. If you wish to continue despite an error, select either the Continue or Throw option.
- **Continue**: Actions are not terminated. Instead, you can select what to do in either event: when there is no error (On Success), or when there is an error (On Error). For example, you might want to display a message box saying whether a page load was successful or not.
- **Throw**: If an error is detected, this option throws an exception that is stored in the Try/Catch action's variable. The Catch part of the Try/Catch action is used to specify what action to take if an error occurs.
occurs. If no error occurs, then the next action is processed. See the section Try/Catch action for details.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

```xml
mt-available-languages()
```

10.10.8 Measure Controls

The Measure Controls action (screenshot below) returns the minimum width in pixels of the specified control kind (button or label, for example) when the string specified in the first setting is the display-text of the control. The action can be used to find control widths of a set of strings and to then use this information in your design. For example, you could find the width of all buttons in a column and then set the width of this column according to the largest button-width.

![Screenshot of Measure Controls action](image)

**Text strings of controls**

Since the control width is dependent on the size of its text, you must provide in the first setting the string that you want to place on the control. If you want to find out the widths of multiple controls, each having a different text, then supply all these texts as the string items of a sequence. The return value will be a sequence containing the widths of each string. This sequence will be stored in the `MT_MeasureControls` variable. For example, in the screenshot above, we have supplied a sequence of two strings: `("Altova Product", "Go to Product Description")`. These strings do not actually have to be present on controls in the design. The return value will contain two number values, which are respectively the widths of the two strings. You could add any string to the sequence to find out the control width for that string.

**Text parameters**

The second setting must be an XPath map expression that specifies the parameters for which the width calculation is made. The parameters include such information as whether the control being measured is a label or a button, or whether the text size is small or medium, or whether the text is bold or not. The keys of the map and their allowed values are listed below. The only mandatory key is "Control Kind". If a key is not specified, then its default value is used. In the screenshot above, for example, the width is being measured for buttons on which the text appears with the default values of the respective parameters. The XPath expression is:

```xml
map("Control Kind":"Button")
```
The Parameters argument is a key-value map that defines the properties of the control. The available keys and their values are listed below. The integer that is returned is the minimum width, in pixels, of the control when the submitted Text string (first setting) is displayed with the properties specified in the Parameters argument. This value can then be used to calculate and specify other properties related to the control, such as the widths of table columns in which the control appears.

The return value and the "MT_MeasureControls" variable
The action returns a sequence of numbers, which is automatically stored in the MT_MeasureControls variable. You can use MT_MeasureControls in the XPath expressions of actions. Since the items in the returned sequence are numbers, you can use the max() function to find out the largest control width (for example: max(MT_MeasureControls)). Another useful function is the serialize() function, which can be used to display the complete sequence.

Points to note
Note the following points:

- For each action, you can find out the widths of one control kind only. To find out the widths of another control kind, call the action a second time with a new Control Kind parameter value.
- This action provides the same functionality as the mt-control-width() function. A key difference is that the function cannot be used for web-client rendering, whereas the action can be used for web clients.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.10.9  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

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.10.10  Log Message

The Log Message action (see screenshot below) enables a customized message to be logged on the server or client during the execution of an action. This helps to analyze the app's behavior during an action. A message can be logged at any point during the execution of an action, based on the selected severity: info, warning, error.
The action takes as its input:

- The location where the message is to be logged: server or client
- The severity of the error: info, warning, error
- The message to be logged, which must be entered as an XPath string

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

`mt-available-languages()`
10.11 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.

10.11.1 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 \texttt{current-date-no-TZ()}).

There are a few points to note when specifying the update node/s and update value/s:

- Importance of context node for relative paths. (See below for details).
- The target node for the update can be referenced with the \texttt{$MT\_TargetNode} variable. (See below for details).
- If the source node is an element with mixed content (text and elements), then only the text content of the mixed-content element is used for the update. The text content of descendant elements is ignored. (See below for details).
- Multiple target nodes can be specified via an array. (See below for details).

Importance of context node for relative paths

If the node to be updated is specified as a relative path, then only those nodes are updated that are descendants of the context node. To update a descendant node of sibling elements, an absolute locator path will need to be used. This is explained in more detail below.

- In the screenshot below, notice that the context node (indicated in the Context field) is the Row element of the DB. The context node is the node within which the control (for which the action is being defined) is located (or with which the control is associated).
- The @Updated attribute that will be updated is therefore the @Updated attribute of that particular Row element. What this means is that when a Row element's control event is triggered, then the @Updated attribute of only that Row element is updated, in this case with the current date.
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:

```xml
$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 page sources.

Referencing the specified target node with $MT_TargetNode

After a target node for the update is defined, you can reference this node with the variable $MT_TargetNode. For example:

Update Node: @Updated
Update Value: concat(current-date-no-TZ(), '-ID-', $MT_TargetNode/../@id)

would give an update value that would have the @id attribute value of the current Row element suffixed to the current date, as in the screenshot below.
Now, if the update nodes were the @Updated attributes of all Row elements, and the XPath expression for the update value were the same as in the previous example:

Update Node: $DB2/DB/RowSet/Row/@Updated
Update Value: concat(current-date-no-TZ(), '-ID-', $MT_TargetNode/../@id)

then the @Updated attribute of each Row element would have a value that would have the @id attribute value of its own Row element suffixed to the current date.

Modifying and updating nodes with $MT_TargetNode
The $MT_TargetNode variable can be very useful if you want to modify and update nodes in one joint action. An example is shown in the screenshot below.

The action in the screenshot works as follows:

- The nodes to update are all the attributes of the current element. These nodes are specified by the XPath expression of Update Node(s): @*
- As each attribute is processed it becomes the subject of $MT_TargetNode
- The Result setting specifies the updated value of the node, which, in the example above, is the original value of the attribute (obtained via $MT_TargetNode)—modified in that its leading and trailing spaces are removed by the trim-string XPath function.
- The overall result is that each attribute will have any leading or trailing spaces removed

Source elements with mixed content
If an element with mixed content (text and element/s) is located with an XPath locator expression, then the text
content of the mixed-content element only is returned. The text content of descendant elements is ignored.

This is best explained with an example of the Update Node(s) action. Consider the Update Node(s) action defined in the screenshot below.

If the XML tree had the following structure and content:

```xml
<Element1>
  <source>AAA
    <subsource>BBB</subsource>
  </source>
  <target></target>
</Element1>
```

Then the target element would be updated with the text content of the mixed-content element source, while ignoring the content of its child element subsource. The node named target will be updated to `<target>AAA</target>`.

**Note:** If you wish to include the text content of descendant node/s, use a string function. Using the XML example above, for instance, the expression `string($XML1/Element1/source, '')` will return "AAABBB".

**Note:** Charts use the XPath compliant method of serializing: When a mixed-content element is located using an XPath locator expression, then the text content of descendant elements is also serialized.

### Updating multiple nodes via XPath arrays

You can update multiple nodes (the target nodes) by locating these nodes via an XPath array. The update values must then also be submitted as an array, one which has the same size as the target-node array. The update happens 1:1 and member-wise: the first member of the value array updates the first member of the target array, and so on. For example:

Update Node : $XML1/User/Message/[@subject, @date, @senderID]

Note the following points:

- The $MT_TargetNode variable is set to the array that identifies the target nodes. You can check whether the variable contains an array as follows: `if($MT_TargetNode instance of array(*)) then...`
- The evaluation and mapping of the arrays are carried out first; the assignments are executed as a group subsequently. This means that the value of a target node can be used as an input.
Each member of the target array is handled separately. As a result, even if one member of the target array produces an error (either itself or while its value was evaluated), the updates of other target nodes will continue. Appropriate error messages, however, are generated.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.11.2 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:

```xml
<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 page source on the page. For example:

```xml
$XML2/Row
```

XQuery's computed node constructors can also be used, for example:

```xquery
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.

```xml
<MyElement-01 myatt="value">Element Content</MyElement-01>,
MyElement-02 {"Element Content"},
MyElement-03 {element MyElement-04 {"Element Content"}},
attribute myatt {"value"}, MyElement-06{}
```
Removing inserted nodes from their original locations

If the inserted node/s are obtained from one of the project's page sources, you can delete the node/s from their original location by selecting the *Remove inserted node(s) from their current location* check box. If new nodes are constructed directly—that is, without reference to the project's page sources—then selecting this option will have no effect on the page sources.

The `$MT_TargetNode` variable

In the Insert Node(s) definition, the node before which the insertion is carried out is automatically saved in MobileTogether Designer's built-in variable `$MT_TargetNode`. This variable can then be used in the second XPath expression of the definition, as has been done in the screenshot below.

```
@ OnFinishEditing
? If Selection="new" X
| Then
| Insert before Node Product[1]
| new Node(s) element {concat("MyNew", name($MT_TargetNode))} {"Element Content"}

| remove inserted node(s) from their current location
```

The second XPath expression that creates the new node using the same name as the target node ($MT_TargetNode) as part of the new node's name.

```
element {concat("MyNew", name($MT_TargetNode))} {"Element Content"}
```

The result of the Insert Node(s) action defined above is shown in the screenshot below.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

### 10.11.3 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 node/s, select the appropriate radio button in the action's definition.

New nodes

New nodes can be entered as direct XML constructors as in the screenshot above:

<MyElement-01 myatt="value">Element Content</MyElement-01>

This appends the element MyElement-01 after the last Product element, as in the screenshot below.
You can also use an XPath locator expression to append a node (and all its descendants) from a page source. For example:

$XML2/Row

XQuery's computed node constructors can also be used, for example:

```xml
<element MyElement-01 {xs:string("Element Content")}>
    <attribute myatt="value"/>
</element>
```

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).

```xml
<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 page sources, you can delete the node/s from their original location by selecting the Remove appended node(s) from their current location check box. If new nodes are constructed directly—that is, without reference to the project's page sources—then selecting this option will have no effect on the page sources.

A good example of how to use the Remove appended node(s) from their current location option is the sorting of nodes. Suppose we have a tree with this structure: $XML1/products/product/@name. We want to sort the product nodes on the basis of their @name values. We can do this with the Append Node(s) definition shown in the screenshot below.

- 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.

```
  Appending to Node current()
  new Node(s) {element(concat("MyNew", name($MT_TargetNode/*[last()])))
    {Element Content}

  as first child  as last child
  remove appended node(s) from their current location
```

The second XPath expression uses the target node ($MT_TargetNode) to find the target node's last child element and then uses the name of that child element to build the name of the new element.

```
element (concat("MyNew", name($MT_TargetNode/*[last()])))
  {Element Content}
```

The result of the Append Node(s) action defined above (when the target node is `$XML1/Products`) is shown in the screenshot below.
MobileTogether extension functions

MobileTogether provides a range of *XPath extension functions* that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the *Message Box* action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic *MobileTogether Extension Functions*.

10.11.4 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 `$XML1/Products/Selection` element. The context node is `Products`. The action deletes the child `Product` element that has content, which when converted to lowercase matches the (lowercase) content of the `Selection` element. In the screenshot below, `mobiletogether` has been selected in the combo box and becomes the value of the `Selection` element. The `Product` element containing the text "MobileTogether" has been deleted.

MobileTogether extension functions

MobileTogether provides a range of *XPath extension functions* that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the *Message Box* action. If a function is especially relevant to this action, it is listed below. For a
full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.11.5 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 appends a *subnode* named **Body** as a last child from a *source node* named **ArticleOriginal** to a target named **ArticleCopy**. The screenshot below shows the page sources during a simulation, after the action has been executed. Notice that the **Body** element of **ArticleCopy** has been replaced by the **Body** element of **ArticleOriginal**.
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.

MobileTogether extension functions
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be
used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.12 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
- Break Loop
- Let
- Update Variable
- 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.

10.12.1 If-Then

Sets the action to be carried out if the IF condition evaluates to true. The action to be carried out is dropped as a child of the THEN clause.

The condition in the definition above tests whether the current element has an attribute called company that has a value of Altova. If true, the Altova website URL is opened.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.12.2 If-Then-Else

Sets the actions to be carried out for both the true and false evaluations of the IF condition. The action to be carried out for a true evaluation is dropped as a child of the THEN clause. The action to be carried out for a false evaluation is dropped as a child of the ELSE clause.

The condition in the definition above tests whether the current element has an attribute called company that has a value of Altova. If true, the Altova website URL is opened. If false, the solution is exited.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.12.3 Loop

The Loop action (see screenshot below) iterates over a sequence of items that you define using the For Each setting. Within the loop you can then define a set of actions to perform during every iteration. For example, in the screenshot below, for each iteration, an Append to Node action is carried out. Also inside the loop, a Break Loop action is specified if a certain condition is fulfilled.

The sequence over which the loop iterates is defined by the XPath expressions of the For Each setting. The key points to note are given below.

- **For Each**: Can be a sequence named in the XPath expression (for example: `1 to 7`) or one obtained from an XML tree (for example: `$XML1/Products/Product` selects a sequence of all `Product` elements in the `$XML1` tree; see screenshot above). If the loop does not contain a Break Loop action, then the loop is exited when all iterations have been completed.
- **Loop variable**: The loop variable is the variable that holds the item of the sequence that is currently being iterated over. The loop variable is identified by a name, which you enter by first double-clicking after the `$` sign and then typing the name. In the screenshot above, the loop variable is named `MyLoop`. It is referenced like any other XPath variable, that is, with a `$` sign before its name (`$MyLoop`). The variable will be in scope within the loop; this means that you cannot reference the variable in an XPath expression that is outside the loop. In the screenshot above, the loop variable is referenced in the New Node setting of the Append to Node action. This is a valid reference since the Append to Node action has been created within the loop; the variable is therefore in scope at this point. The `$MyLoop` variable shown in the screenshot above will contain the `Product` node that is currently being iterated over.

**Note**: If the loop variable uses nodes from a page source tree, then this tree is locked and cannot be modified by the actions inside the Loop action. In our example above, the tree `$XML1` will be locked while the
loop is being processed. The modifications are happening in another page source tree ($XML2): the new nodes that are added with the Append to Node action are added to $XML2. If you want to modify the tree being iterated over, you can do this as follows: Instead of iterating directly over the nodes of the tree, iterate over a number sequence that is tied to the node sequence in the tree. For example, instead of iterating over the sequence of Product nodes in the screenshot example above, we can iterate over a range of numbers that is tied to the node sequence. The XPath expression of the For Each setting can be changed from $XML1/Products/Product to for $i in 1 to count($XML1/Products/Product) return $i. It is a sequence of numbers that is now being iterated over. (The current Product node within the loop can be accessed with the XPath expression: $XML1/Products/Product[$i]).

The actions defined in the screenshot above duplicates the first five Product elements in another tree. When the page is loaded, the Loop action iterates over the $XML1/Products/Product elements. During each iteration, the current Product node is stored in the MyLoop variable. This Product node (in the $MyLoop variable) is then added as the last child of the $XML2/Products node. The loop continues till the fifth Product element has been copied from $XML1/Products to $XML2/Products. See screenshot below.

The loop is exited either when all iterations have been completed or a condition is set under which a Break Loop action is executed (see next topic).

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
10.12.4 Break Loop

The Break Loop action (see screenshot below) is used within a loop and is used to exit the loop. If you want the loop to be exited when a specific condition is fulfilled, you must specify this condition in, say, an If-Then action, and place the Break Loop inside the If-Then action (see screenshot below). If you do not place the Break Loop action within a condition, then the loop will be broken directly the Break Loop action is encountered.

The actions that are performed in the loop that is shown in the screenshot above are described in the Loop topic. Here, we focus on the Break Loop action. In our example (see screenshot above), the Break Loop action is executed when the number of appended nodes in $XML2 reaches 5. When the iteration over the sixth Product element of $XML1 starts, the condition condition defined in the If clause is checked. The XPath expression for the condition is: \[ \text{count}($XML2/Products/Product) = 5 \], which now evaluates to true(). As a result, the Then clause is executed and the loop is exited. See screenshot below.
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the `Message Box` action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

### 10.12.5 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.

Also see the description of the Loop action *(previous topic).*
Note: The variable defined in a Let action is in scope only within that Let action. This means that it can be used only in child actions of the Let action.

Note: If a variable contains a nodeset and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. However, this does not apply if only the values of variables in a nodeset have been modified.

The XPath option
The XPath option enables you to enter a static value or to generate a dynamic value using XML tree nodes. For example, in the screenshot below, the values of two nodes are multiplied (circled in red). The resulting value will be the value of the variable ($area). The variable has then been used to update the content of an XML tree node.

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.

![RectangleArea](image)

**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.

![OnPageLoad for page 'Geometry'](image)

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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.12.6 Update Variable

The action updates the value of an existing variable that you identify by naming it in the action (source in the screenshot below). The new value is the result of evaluating the XPath expression that you enter. The Update Variable action therefore enables you to modify the value of a variable that was defined when the variable was initialized.

The variable to be updated can be any user-defined variable that is in scope at the point when the Update Variable action is triggered. These variables include: user-defined global variables; variables defined in the Let and Try/Catch Exceptions actions; variables in Action Groups; the parameters of a sub page; and the parameters and variables of control templates. (Note that the Update Variable action cannot be applied to Loop variables.)

At run time, if no variable with the given name is found, then an error message to this effect is generated together with a list of variables that are in scope at that point and may be modified.

Note: Application-defined variables (such as Dynamic Local Variables and Static Global Variables) cannot be modified by the Update Variable action.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.12.7 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:

```xml
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.

**MobileTogether extension functions**
MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a
full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.

10.12.8 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.

**Note:** If a variable contains a nodeset and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. However, this does not apply if only the values of variables in a nodeset have been modified.
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.

MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, mt-available-languages() returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.
### 10.12.9 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](#)).

![Try and catch server connection errors screenshot](#)

**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 ([see screenshot below](#)) action, (ii) action/s for the OnServerConnectionError event, (iii) a MobileTogether message about the error, following which the workflow is resumed.

### MobileTogether extension functions

MobileTogether provides a range of [XPath extension functions](#) that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the [Message Box](#) action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic [MobileTogether Extension Functions](#).

### 10.12.10 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 ([see screenshot below](#)). 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 ([see screenshot below](#)) 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.
MobileTogether extension functions

MobileTogether provides a range of XPath extension functions that have been specifically created for use in MobileTogether designs. Some functions can be particularly useful with specific actions. For example, `mt-available-languages()` returns the languages in which the solution is available and could, for example, be used with the Message Box action. If a function is especially relevant to this action, it is listed below. For a full list of extension functions and their descriptions, see the topic MobileTogether Extension Functions.


10.13 **Action Groups**

An Action Group defines a sequence of actions to execute. If the same sequence of actions has to be executed at various points in the workflow, then it is efficient to create 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.

### 10.13.1 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).

![Action Groups pane](image)

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.

![Manage Group Actions dialog](image)
**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.

### 10.13.2 Action Groups for Reusing Actions

Action Groups can be used to group a set of actions for reuse. Set up an Action Group for this purpose as follows:

1. Access the Action Group in order to edit it.
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.

   ![UpdateUTCTime Action Group](image)

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.

![Screenshot of Execute Action Group action](image)

Note the following points:

- All the actions in the Action Group will be carried out, in the specified order, when the event is triggered.
- The Action Group can be used to execute the same set of actions at multiple points in the workflow.
- Note that the Execute Action Group action has a combo box that enables you to select any of the Action Groups defined in the project (instead of the current Action Group).
- You can also use an XPath expression to select an Action Group. The expression must evaluate to the name of the Action Group you want.
- You can click the Action Group's Edit button to edit the currently selected Action Group.

### 10.13.3 Action Groups with Parameters

Parameters in Action Groups work as follows:

- You declare parameters in an Action Group, and then define actions (within this Action Group) that use these parameters.
- The values of parameters are passed to the parameters when the Action Group is called via the Execute Action Group action.

Declaring parameters in Action Groups, and defining actions that use these parameters

In the Action Group, declare the parameters that are needed to generate the required result. Do this by clicking the Add Parameter icon (see screenshot below). After a new parameter has been added, double-click to the right of the parameter's $ symbol, and enter the name of the parameter.
In the screenshot below, we have an Action Group named RectangleArea, in which we have declared two parameters named $length$ and $width$. The parameters are not marked as Optional. This means that when the Action Group is called at runtime, the action group must receive values for both these parameters (see Supplying parameter values with Execute Action Group below); otherwise an error is reported. (For information about the Action Group check box, see the section Action Groups with Action-Group Parameters.).

The Action Group in the screenshot above 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 MessageBox action, which displays a message box containing the value obtained by multiplying the two parameters $length$ and $width$.

Note that, in the Action Group, we declare parameters and define actions that use these parameters. The values of the parameters are supplied at runtime, via the Execute Action Group action.

Variables in Action Groups
You can also define variables in Action Groups, that is you can declare variables and define a value for each. Note the following key points about variables:

- They are in scope within the Action Group, and can only be used within the Action Group
- The definition of the value of a variable can use parameters and variables that have been declared earlier in the list of variables. For example: If an Action Group has parameters $a$, $b$, $c$, and variables $x$, $y$, $z$ (in that order), then the variable $y$ can use the following parameters and variables to generate its value: $a$, $b$, $c$, $x$, (but not $z$).

Supplying parameter values with Execute Action Group
You can define an Execute Action Group action on an event, and in the action you can define the parameter values to pass to the selected Action Group. When the event is triggered, the parameter values are passed, and the Action Group uses these values to carry out the actions defined in the Action Group.
To create an Execute Action Group action, drag and drop the relevant Action Group (from the Action Groups pane) 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 OnButtonClick event. We did this by dragging the RectangleArea Action Group below the On Click event. The currently selected Action Group to execute can be changed in the Execute Action Group combo box (circled in red in the screenshot below). Click the Additional Options icon to open the selected Action Group. Alternatively, you can use an XPath expression to select the Action Group. Using an XPath expression enables you to select the Action Group dynamically (for example, by getting the name of the Action Group from a page-source node). The XPath expression must return a string that is the name of an Action Group defined in the project.

If parameters have been declared in the currently selected Action Group, then these parameters are displayed in the Execute Action Group action. This is where you supply the values to pass to parameters at runtime. In the screenshot above, it can be seen that the RectangleArea Action Group has two parameters: $length and $width. We have entered simple static values (3 and 4) in the XPath expressions that are used to generate the values of these two parameters. But you can also obtain values from XML tree sources dynamically, or specify complex XPath calculations. If, in the Action Group, the parameters were declared as mandatory (not optional), then they will be displayed here in red if no values are supplied.

At runtime, the following happens:

1. When the event is triggered, the parameter values (as specified in the Execute Action Group action) are passed to the respective parameters in the Action Group.
2. The Action Group's actions are processed. Where these use the Action Group's declared parameters, the supplied parameter values are substituted.

Types of parameter values
In the Execute Action Group action, parameter values can be of the following types, either entered directly or generated dynamically (by using XPath expressions):

- Atomic values: for example, strings such as "Altova" or numbers such as 1 or 2.56
- Arrays: for example, [(2010,2019), ("StartYear","EndYear")]
- Maps: for example, map{"StartYear":2010, "Offices":{"Boston","Vienna"}}

10.13.4 Action Groups with Action-Group Parameters

Action Groups can use two types of parameters:
Simple parameters, which are described in the section `Action Groups with Parameters`. They are indicated by the dollar symbol $ in front of their names.

- **Action-Group parameters**, which take Action Groups as their values. They are indicated by the percent symbol % in front of their names.

In the screenshot below, $length$ and $width$ are simple parameters, while %RectangleProperty% is an Action-Group parameter. To create a parameter as an Action-Group parameter, create it as a simple parameter and then check its Action Group check box (see screenshot below).

**How Action-Group parameters work**

An Action-Group parameter is a parameter whose value is an Action Group that declares no parameter. This enables the called Action Group to be treated like a function that performs MobileTogether tasks. Action-Group parameters work as follows:

- In an Action Group, declare Action-Group parameters. These parameters can be used in actions of the Action Group. In the screenshot above, for example, the Calculate Action Group declares the RectangleProperty Action Group to be one of its parameters. The RectangleProperty Action Group thus becomes an Action-Group parameter. It is then used inside the Calculate Action Group.
- The values of all parameters of an Action Group (including those of any Action-Group parameter) are supplied when the Action Group is called. This happens when the Execute Action Group action is executed on that Action Group. For example, when we call the Calculate Action Group (shown in the screenshot above) via an Execute Action Group action, we pass it the values of the Action Group's three parameters (see screenshot below).
Usage example

Here is a simple example to show how Action-Group parameters can be used. Let us say we want to calculate the area, diagonal length, or perimeter of a rectangle from a user-given length and width (see screenshot below). The user can select which rectangle property to calculate.

For each of the three properties, we can create an Action Group that declares no parameters. In effect, these three Action Groups are used as functions to calculate the respective rectangle property, as shown in the screenshot below.

We now create a fourth Action Group (named Calculate in our example; see screenshot below) that will use one of the other three Action Groups as required. To select the Action Group we want, we create an Action-Group parameter (named %RectangleProperty in our example). Note that the Calculate Action Group also declares the two simple parameters ($length and $width) that are used to calculate the area, diagonals, and perimeter.
All that we need to do now is to call the *Calculate* Action Group via an Execute Action Group action and, in the call, supply the values of the *Calculate* Action Group’s three parameters. The Execute Action Group action is carried out ideally when some event is triggered. For example, in the screenshot below, the action is carried out when a combo box is edited.

What happens when the Execute Action Group action is carried out is this:

1. The *Calculate* Action Group is called. Values of its $\text{length}$ and $\text{width}$ parameters are passed from nodes of the $\text{SPERSISTENT}$ page source (see screenshot above). The values of the $\text{RectangleProperty}$ Action-Group parameter must be an Action Group in the project and can therefore be selected via the combo box; any one Action Group (corresponding to a chosen rectangle property) can be selected. In the screenshot of our example (above), we have selected the *Area* Action Group (in the event that the $\text{SPERSISTENT/Root/RectangleProperty}$ node contains the string “*Area*”).
2. The *Calculate* Action Group is now executed with the parameter values passed to it. If you look at its screenshot above, you will see that it defines a variable named $\text{result}$ and gives it a value that is the return value of the Action Group specified by the $\text{RectangleProperty}$ parameter. This variable will therefore contain the value of the selected rectangle property (area, diagonal length, or perimeter).
3. The *Calculate* Action Group now updates a node in the $\text{SPERSISTENT}$ tree with this result.
Details of the Action Group definitions

- **Calculate** (see screenshot above): Declares two simple parameters ($\text{length}$ and $\text{width}$) and an Action-Group parameter ($\text{RectangleProperty}$). 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 that is set to Action Group Result. The Let action defines a variable called $\text{result}$, which is defined to take as its value the result of executing the selected Action Group.

- **Area** (screenshot below): This Action Group contains a single Return action that contains the product of the two simple parameters $\text{length}$ and $\text{width}$. This Action Group contains no parameter; it can therefore be the value of $\text{RectangleProperty}$. XPath to calculate area: round($\text{width} \times \text{length}$, 3).

- **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 $\text{length}$ and $\text{width}$; XPath: round(math:sqrt($\text{width} \times \text{width} + \text{length} \times \text{length}$), 3)).

- **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: round(2*$\text{width} + 2*$length), 3)).

In the description of the four Action Groups above, note the following:

- The Calculate Action Group contains three parameters: two simple parameters and one Action-Group parameter.
- The other three Action Groups (Area, Diagonals, Perimeter) declare no parameter.
- Each of the three Action Groups Area, Diagonals, Perimeter has a Return action that performs a calculation. The returned value is the Action Group Result.
- Each of the three Action Groups Area, Diagonals, Perimeter can be set as the value of the Action-Group parameter $\text{RectangleProperty}$, and would return their respective Action Group Result when processed.

**Note:** The example above is deliberately simple; it aims to show the mechanism behind Action-Group parameters. Action-Group parameters, however, are best used with dynamic content and to execute complex actions.

**Processing of Action-Group parameters**

An Action-Group parameter takes an Action Group as its value. This Action Group can be processed as a parameter value in two useful ways:
- It can provide a result—the **Action Group Result** 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. This kind of usage enables you to reuse an Action Group to perform repetitive tasks. The repetitive task can be defined in an Action Group, and the Action Group can be executed at different points during solution execution.

You could of course combine both sets of actions.

### 10.13.5 Variables and Action Group Results

You can set up an Action Group to return a value—the Action Group Result. (For example, in the screenshot below, we have declared two parameters, and then multiplied them in a **Return action** 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**.

**Note:** If a variable contains a nodeset and the nodeset is modified during processing, then the variable is invalidated and cannot be used subsequently. However, this does not apply if only the values of variables in a nodeset have been modified.

The steps to carry out for this usage are as follows:

1. In the Action Group, declare any parameters that may be needed to generate the required result. Do this by clicking the **Add Parameter** icon (see screenshot above). After a new parameter has been added, double-click to the right of the parameter's $ symbol, and enter the name of the parameter. In
the screenshot above, we have an Action Group named RectangleArea, in which we have declared two parameters named $length and $width. Note that: (i) the parameters have been declared but no values have been defined for them; (ii) the parameters are in scope only within the Action Group, and cannot be used outside it. If a parameter is defined as optional, then it is not an error if it is not used in the definition of the variable (see Step 3 below).

2. Add a Return action. 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.
11 Design Components

This section describes a range of design objects that can be placed on a page, most of which are controls that can be placed on a page, as well as advanced features that can be defined for a page and/or the project.

- **Tables**, 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
- **Control Templates**, which describes how to create a template that can be used at multiple locations 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.
- **Solutions for Authenticated Users**, which describes how to set up authentication so users can be directed to a solution on another MobileTogether Server without having to log in a second time
- **Hyperlinking to Solutions**, 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 comprised 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 page 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

### 11.1.1 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 page source
- A page control (with or without a link to a page source node)
- A nested table

Cell content is typically a page control. 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.)

### 11.1.2 Repeating Tables

Repeating tables work as follows:

- A repeating element in the page source is associated with the repeating table (see examples below).
- 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 page 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 page source contains, say, First, Last, and Phone child elements. The Person element can occur multiple times (its instances). We would like to create a table like this:
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: \( \text{count(preceding-sibling::*\*\*)+1} \). The design would look something like this:

![Person ($XML2)](image)

The generated table would look something like this:

<table>
<thead>
<tr>
<th></th>
<th>Basil</th>
<th>Brown</th>
<th>1234567</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mary</td>
<td>Gold</td>
<td>4567890</td>
</tr>
<tr>
<td>2</td>
<td>Daisy</td>
<td>White</td>
<td>7890123</td>
</tr>
</tbody>
</table>

**Note:** A data stream can be generated from an XPath/XQuery expression and can be used as a data source. However, this kind of data source is created for current use only and cannot be accessed as a page source for use elsewhere in the document.

**Difference between a repeating table and a 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 generated 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 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 page source.
Creating a repeating table

Set up a repeating table as follows:

1. In the New Table dialog that appears when the Table control is dropped in the design (screenshot below), check *Table will be repeating* to make the table repeating. Note that it is the table that repeats for every instance of a repeating data row.

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 page 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 page source node), or even page source nodes. When a page source node is dropped into a cell, the data in that cell will be editable. In the screenshot below, four controls have been added (from left to right): a label, edit field, label, and edit field.

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.

### 11.1.3 Dynamic Rows

Dynamic rows work as follows:

- A dynamic row is associated with a repeating element in a page 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 page 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 page 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 page source changes, then the number of rows in the table is modified automatically.

<table>
<thead>
<tr>
<th></th>
<th>&lt;First&gt;</th>
<th>&lt;Last&gt;</th>
<th>&lt;Phone&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Person&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 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 generated 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 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 page 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 page source node), or even page source nodes. When a page source node is dropped into a cell, the data in that cell will be editable. In the screenshot below, four controls have been added (from left to right): a label, edit field, label, and edit field.

   ![Screenshot](image)

   This dynamic row produces the following structure in the MobileTogether solution.

   ![Screenshot](image)

   For information about spanning dynamic columns, see the section **Row/Column joining and spanning**.

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Vienna</td>
</tr>
<tr>
<td>21</td>
<td>Munich</td>
</tr>
<tr>
<td>22</td>
<td>London</td>
</tr>
<tr>
<td>23</td>
<td>Paris</td>
</tr>
<tr>
<td>24</td>
<td>Boston</td>
</tr>
<tr>
<td>25</td>
<td>Tokyo</td>
</tr>
<tr>
<td>26</td>
<td>Moscow</td>
</tr>
</tbody>
</table>
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.

11.1.4 **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 page 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).

<table>
<thead>
<tr>
<th></th>
<th>&lt;First&gt;</th>
<th>&lt;Last&gt;</th>
<th>&lt;Phone&gt;</th>
</tr>
</thead>
</table>

- **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 page source must be the same for all rows. For example, in the table below, each of the four `week` elements (each corresponding to a row in the table) contains exactly three `day` elements (the columns of the table). If any `week` element contains some other number of `day` elements (the columns of the table), then the table cannot be correctly drawn. Note also that, typically: (i) column elements occur within row elements, both in the page source and in the table design; (ii) the names of row elements are the same and the names of column elements are the same. However, neither of these two points is a necessary condition for building this kind of dynamic table: (i) column elements can occur outside row elements, and (ii) the names of rows/columns can be different.

<table>
<thead>
<tr>
<th></th>
<th>&lt;day&gt;</th>
<th>&lt;day&gt;</th>
<th>&lt;day&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;week&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;week&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;week&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;week&gt;</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**: A data stream can be generated from an XPath/XQuery expression and can be used as a data source. However, this kind of data source is created for current use only and cannot be accessed as a page source for use elsewhere in the document.

**Example: dynamic columns for days inside dynamic rows for weeks**
The screenshot below shows a `calendar` element that contains four `week` elements, with each `week` element containing seven `day` elements. We can make a table containing dynamic rows for `week` elements and dynamic
columns (inside each `week` element) for the `day` elements. Notice that in the data structure (i) the `day` elements are inside the `week` elements, and (ii) the number of `day` elements inside every `week` element is the same: seven.

```
<?xml version="1.0" encoding="UTF-8"?>
<calendar>
  <week id="w1">
    <day id="D1"/>
    <day id="D2"/>
    <day id="D3"/>
    <day id="D4"/>
    <day id="D5"/>
    <day id="D6"/>
    <day id="D7"/>
  </week>
  <week id="w2">
    <day id="D1"/>
    <day id="D2"/>
    <day id="D3"/>
    <day id="D4"/>
    <day id="D5"/>
    <day id="D6"/>
    <day id="D7"/>
  </week>
  <week id="w3">...</week>
  <week id="w4">...</week>
</calendar>
```

Creating dynamic columns inside dynamic rows

Drag a table control into the design and drop it at the location where you want to create it. In the New Table dialog (screenshot below), select the options for dynamic rows and dynamic columns. Select the number of rows and columns that should be repeated for each 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 page source nodes that will provide the data for the table's rows and columns. The element associated with the column must, in the page source, be contained within the element associated with the row. In the design, however, notice that the row field is placed within the column field.
Also note the XPath expressions that are used to associate the row and column fields with page source nodes. The expression that selects the element for the row field must select all instances of the corresponding element. The XPath expression in the screenshot above selects all the week child elements of the calendar element: $XML1/calendar/week. The XPath expression for the column field, however, must select only the column element within the current row. So, an XPath expression such as $XML1/calendar/week/day would not work because it would select all the day child elements of all week elements. Note also that the XPath context node for the column field is the element that is associated with the row. In our example, the context node of the column field is week. As a result, an XPath expression of day would select the day child elements of the current week element.

In the design there is a single cell located at the intersection of the row and column fields. The context node of this cell is the element corresponding to the row field (in our example, the week element). XPath expressions in this cell must be constructed within this context. When the table is created, then within each “row element” (the week element in our example), a cell is created for each column. The XPath expression is evaluated for each column’s cell within the context of the current row. To reach the contents of the element that corresponds to the column field, a special variable is available: $MT_TableColumnContext, which, at runtime, contains the element that corresponds to the current column (in our example the current day element within the current week element). All this is best explained with an example. In the screenshot above, the cell in the design that is at the intersection of the row and column fields contains a label control. 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:

```
<table>
<thead>
<tr>
<th></th>
<th>W1D1</th>
<th>W1D2</th>
<th>W1D3</th>
<th>W1D4</th>
<th>W1D5</th>
<th>W1D6</th>
<th>W1D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2D1</td>
<td>W2D2</td>
<td>W2D3</td>
<td>W2D4</td>
<td>W2D5</td>
<td>W2D6</td>
<td>W2D7</td>
<td></td>
</tr>
<tr>
<td>W3D1</td>
<td>W3D2</td>
<td>W3D3</td>
<td>W3D4</td>
<td>W3D5</td>
<td>W3D6</td>
<td>W3D7</td>
<td></td>
</tr>
<tr>
<td>W4D1</td>
<td>W4D2</td>
<td>W4D3</td>
<td>W4D4</td>
<td>W4D5</td>
<td>W4D6</td>
<td>W4D7</td>
<td></td>
</tr>
</tbody>
</table>
```

- Each week element in the page source is displayed in a row (W1 to W4).
- Each day element within a week is displayed within the appropriate column (D1 to D7).
- The concatenation of the two IDs is executed separately for the 28 cells from W1D1 to W4D7.

For information about spanning dynamic columns, see the section Row/Column joining and spanning.
11.1.5 Table Properties

Certain properties can be set for a table's cells, columns, and rows, as well as for the entire table. The screenshot below left shows the 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.
<table>
<thead>
<tr>
<th>Styles &amp; Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Table Cell</strong></td>
<td></td>
</tr>
<tr>
<td>Background Color</td>
<td></td>
</tr>
<tr>
<td>Width</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td></td>
</tr>
<tr>
<td>Style</td>
<td></td>
</tr>
<tr>
<td>Browser CSS Class</td>
<td></td>
</tr>
<tr>
<td><strong>Table Column</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Table Row</strong></td>
<td></td>
</tr>
<tr>
<td>Visible</td>
<td></td>
</tr>
<tr>
<td>Background Color</td>
<td></td>
</tr>
<tr>
<td>Width</td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td></td>
</tr>
<tr>
<td>Style</td>
<td></td>
</tr>
<tr>
<td>Browser CSS Class</td>
<td></td>
</tr>
<tr>
<td><strong>Table</strong></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>True</td>
</tr>
<tr>
<td>Repeating</td>
<td></td>
</tr>
<tr>
<td>Automatic Append/Delete Controls</td>
<td>true</td>
</tr>
<tr>
<td>Create for each item in</td>
<td></td>
</tr>
<tr>
<td>Maximal Table Width</td>
<td></td>
</tr>
<tr>
<td>Maximal Table Height</td>
<td></td>
</tr>
<tr>
<td>Visible</td>
<td></td>
</tr>
<tr>
<td>Background Color</td>
<td>silver</td>
</tr>
<tr>
<td>Horizontal Alignment</td>
<td></td>
</tr>
<tr>
<td>Margin</td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td></td>
</tr>
<tr>
<td>Border Width</td>
<td></td>
</tr>
<tr>
<td>Border Color</td>
<td></td>
</tr>
<tr>
<td>Border Style</td>
<td></td>
</tr>
<tr>
<td>iOS Table Padding</td>
<td></td>
</tr>
<tr>
<td>Style Sheet</td>
<td></td>
</tr>
<tr>
<td>Browser CSS Class</td>
<td></td>
</tr>
</tbody>
</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 and table row groups. It specifies the number of times the repeating table or repeating table row group is re-created. This number is equal to the number of items in the sequence returned by the property’s XPath expression. The expression can return two types of sequence:

- Nodes from a page source tree. This is an alternative to associating a repeating table (or table row group) with a page source node— an association made by dragging and dropping the node onto the table. An XPath expression of this kind also allows more flexibility in the node selection. For example, the XPath expression $XML1/Offices/Office[@location='US'] returns a sequence of Office nodes that have an attribute of @location='US'. The US filter cannot be applied by using the alternative method of dropping the Office node onto the table. However, this filter can be achieved with the Create for each item in property.

- Items unrelated to the page source tree. For example, in the month October 2014, the expression 1 to subsequence(age-details(xs:date("2014-01-01")), 2, 1) returns a nine-item sequence, namely, the integers from 1 to 9, which is the number of months that have elapsed between 01 January 2014 and a day in October 2014. This is because the basic XPath expression is 1 to X. And X (according to the subsequence function) is the second item of the three-item sequence returned by the age-details function. This latter function returns the “age” of the current day (in our month of October 2014) relative to the input date (01 January 2014) in terms of the three-item sequence years, months, days (which in this case will be 0 years, 9 months and XX days). The second item of the three-item sequence is the number of months in the age, which is 9. Since the returned sequence contains nine items (the range from 1 to 9), the table will be created nine times.

Note: If you wish to preview the results of XPath expressions, run MobileTogether Designer’s built-in simulator (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
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.

|---------|----------|---------|----------|---------|----------|---------|----------|-----|---------|----------|---------|----------|

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.

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.

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 page source, is the parent of the day element). Since the week element repeats, a new table will be created for each week element. If, in the page source, there are multiple day child elements of the week element, and if, in the design, the dynamic columns of the column group are not spanned, then the table (for each week) generated from this design will have as many columns as there are day child elements. If, however, you set the Spans Column Groups 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 [Max Table Height](image) 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](image)). 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](image)). 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](image) for an example.

The [Max Table Height](image) 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 **Max Table Height** property and no **Max 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 **Max 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 **Max 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 **Max 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 borders**

The **Table** menu offers a number of commands to edit table structure. Several table properties are set in the **Styles & Properties Pane**. Additionally, table borders can be conveniently set in the Border Settings dialog (Table | Border Settings).

### 11.1.6 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.

11.2.1 Image Source

The following types of image sources can be used in page designs:

- Binary image files of common formats, such as PNG, BMP, JPG. Images that have binary file sources reference the URL of the image file.
- Base64-encoded strings, which are text encodings of images. Images that have Base64-encoded images access the Base64-encoded string via an XPath expression. The XPath expression typically returns a node containing the Base64 string. MobileTogether reads the Base64 string and generates the encoded image from the string.

Inserting an image in the design

To insert an image in the design, do the following:

1. Drop an Image control 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. MobileTogether reads the Base64 string and generates the encoded image from the string.
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.

<table>
<thead>
<tr>
<th>Embedding URL-sourced images in the design file</th>
</tr>
</thead>
<tbody>
<tr>
<td>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.</td>
</tr>
</tbody>
</table>

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. The example below explains how Base64 images can be used in designs.

See the next section, Base64-Encoded Images, for an example of how to use Base64-encoded images.

11.2.2 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:

  ```xml
  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.

### 11.2.3 Exchangeable Image File Format (Exif)

**Exchangeable image file format (Exif)** is a standard that defines the image formats used by some digital cameras and smartphone cameras. The metadata tags of the Exif standard hold a broad range of information ranging from the image's date-time and geolocation to camera settings and image composition details. When an Exif image is converted to Base64-encoding, the metadata in the image is also converted to Base64, and is available for retrieval.

**Note:** Not all digital cameras or smartphone cameras provide Exif data.

**MobileTogether Designer's Exif functionality**

MobileTogether Designer provides the following Exif-related functionality:

- The **Let User Choose Image action** 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.
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.
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:
  
  ```xml
  $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 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 \( \text{Append Node action} \) 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.

### 11.2.4 Images Chosen by End User

The Let User Choose Image action \( \text{Let User Choose Image action} \) 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 \( \text{Let User Choose Image action} \) 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 \( \text{Load/Save Image to File} \), 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 \( \text{Reload action} \) for the image is required in order to display the new image in the design.

**Example file: UserSelectedImages.mtd**

The design file UserSelectedImages.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 its 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}`.
Design Components

- 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.

```xml
<OnImageClicked>
  <On Click>[
    <If condition>[
      <Append to Node>
        <new Node(s)>[
          <attribute id (if (exists(preceding-sibling::image)) then max(preceding-sibling::image/@id) + 1 else 1)]
        </new Node(s)>
        <as first child>as last child
      <remove appended node(s) from their current location
    </Append to Node>
    <Let User Choose Image>[
      <Target Node>[
        <On OK>
        <On Cancel>
        <On Error>
      </Target Node>
    </Let User Choose Image>
  </On Click>
  </On ImageClicked>
</On ImageClicked>
```

- 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.

  ![On OK screenshot](image)

- 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.

  ![Load/Save Image to File screenshot](image)

- 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).

### 11.2.5 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.

## 11.2.6 Images in Databases

Images in databases can be stored in Base64 format, which is a text format into which binary data can be encoded.
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

11.3.1 Audio Playback

MobileTogether's Audio playback feature enables predefined audio sounds (available on clients) or audio files located on the mobile device or at a remote location to be played back. Audio can be played back on five channels (numbered 1 to 5), and each Audio action is defined for one specific channel.

Audio playback is controlled via the Audio (Playback) action. 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 page 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's project property. Since each of these events apply across the entire project, each event could be triggered by the audio on any channel. The SMT_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 SMT_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.

### 11.3.2 Audio Recording

MobileTogether's Audio Recording feature enables audio to be recorded via the client device's audio recording app and be saved to a file on the client device.

Audio recording is started via the Audio Recording (Start) action (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.

![Screenshot of Audio Recording feature](image-url)
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.
11.3.3 Text to Speech

The Text to Speech feature converts a text string to audio and plays it back. The text string to play back can be specified directly in the Start settings of the Text to Speech action (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 enable further action to be taken at those points in time when these events are triggered.

XPath functions related to Text to Speech

The following text-to-speech-related MobileTogether XPath extension functions 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.

11.3.4 Video Playback

MobileTogether's Video playback feature enables: (i) remote video files to be directly streamed to the client device, and (ii) locally saved video files to be played. Video playback is defined in two steps:

1. A Video control 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 page 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 functions](#) are available:

- `mt-video-get-current-position( VideoControlName as xs:string ) as xs:decimal`
- `mt-video-get-duration( VideoControlName as xs:string ) as xs:decimal`
- `mt-video-height( VideoControlName as xs:string ) as xs:integer`
- `mt-video-width( VideoControlName as xs:string ) as xs:integer`
- `mt-video-is-playing( VideoControlName as xs:string ) as xs:boolean`

You can use these functions in XPath expressions, for example, to specify processing that is conditional on the height/width of the video. Note that before video playback starts, details about the video file are not available. As a result, information such as the height, width, duration and current position will not be known. The corresponding functions should therefore only be used after playback starts.
11.3.5 Video Recording

MobileTogether’s video recording feature enables: (i) the mobile device’s recording app to be started from the MobileTogether solution when an event in the solution is triggered, and (ii) the file of the video recording to be automatically saved at a designated location with a designated file name when the recording is stopped.

The feature enables the user of the mobile device to directly record video with the device’s video recording app (after this has been started by the video recording action). When the user stops the recording, or when the maximum recording duration or recording file size has been reached, the video file is saved to the designated location.

Setting up a video recording

A video recording is defined in a Video Recording action (screenshot below). This action can be set within a sequence of actions to perform when a particular event (page event or control event) is triggered.

In the Video Recording action (screenshot above), you can define the following properties:

- The name and location of the video file to save when the recording has been ended.
- The maximum recording duration or file size. This can be used to restrict the duration or size of the recording.
- The picture quality of the video recording.

For details of these properties, see the description of the Video Recording action.

Useful MobileTogether XPath extension functions

The following MobileTogether XPath extension functions can be useful (for example, to find the location of the recorded file if this has not been specified in the properties of the Video Recording action, or to find the file extension of the video file):

- `mt-last-file-path` gets the file path and file name of the recording. You can use this function not only to determine the location of the recorded-video file, but also to submit this file path as the argument of the next two functions.
- `mt-extract-file-name` gets the file name from the submitted file path.
- `mt-extract-file-extension` gets the file extension from the submitted file path.
Note about simulations
For simulations, you will be able to select a video recording from disk to use instead of the recorded video file, and action handling will continue.

11.3.6 Audio/Video Formats

Given below are links to Internet pages that contain information about the audio and video formats supported by the various client devices that run MobileTogether solutions. We have selected web pages that we find the most useful. However, if the information you require is not available on the pages listed below, please search for alternative sources.

**Note:** MIDI file playback is supported on all client devices except Web browsers. On iOS devices, however, MIDI file playback requires a sound bank file.

### Android

[Android Developer Website: Supported Media Formats](#)

**Audio recording formats**

In general, the supported formats/codecs for audio recording will depend on the device and the OS version. Because of the large number of device-OS combinations that are available it is best not to select a specific format or encoder, and, instead, to use the default format/codec of the respective devices. Note the following restrictions:

- For recording to AMR-NB and AMR-WB audio formats, use only the AMR-NB and AMR-WB codecs, respectively.
- The AAC audio format can be recorded with the AAC-Low Complexity, High Efficiency-AAC, and Enhanced Low Delay-AAC codecs, but it is possible that not all of these codecs will be available or work on a given device.

### iOS

iOS supports many industry-standard video formats and compression standards, including the following:

- H.264 video, up to 1.5 Mbps, 640 by 480 pixels, 30 frames per second, Low-Complexity version of the H.264 Baseline Profile with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in .m4v, .mp4, and .mov file formats
- H.264 video, up to 768 Kbps, 320 by 240 pixels, 30 frames per second, Baseline Profile up to Level 1.3 with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in .m4v, .mp4, and .mov file formats
- MPEG-4 video, up to 2.5 Mbps, 640 by 480 pixels, 30 frames per second, Simple Profile with AAC-LC audio up to 160 Kbps, 48 kHz, stereo audio in .m4v, .mp4, and .mov file formats
- Numerous audio formats, including the ones listed in [Audio Technologies](#)

[Apple Developer Website: Audio Information](#)
[Apple Developer Website: Video Information](#)
Windows

**Audio/Video codecs and formats for Windows**

**Audio recording formats**
WMA and MP3 files are not supported as recording formats on Windows Phone.

Web browser

**Basic overview of playing audio in HTML**
**Basic overview of playing video in HTML**
**Support of audio/video formats in various browsers**

The best supported formats currently are:

- **Audio**: mp3, aac
- **Video**: H.264 (mp4)
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](#)
11.4.1 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.
11.4.2 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.

![NFC Push action screenshot](image)

See the **NFC Push** action for details of how to define the action's settings.

**Note:** NFC data-transmission is not supported on iOS devices.
11.4.3 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.

![OnPushNdefMessageCompleted](image)

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.
11.4.4 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-readable 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.
Push Notifications

A push notification (PN) is a text message that appears on a mobile device and that is related to an app that is installed on the device. When a PN is received on the device, the user does not need to be using the app or even using the device. All that is required is that the device be switched on. PNs thus provide a way for app publishers to communicate directly to users, without having to wait for the app to be started or to get lined up among incoming messages. PNs typically provide information, such as update news related to the app, but can also be used to drive actions, such as accepting invitations, linking to a website, or modifying a database.

In MobileTogether, a PN is sent from a MobileTogether solution on one device and is received by the same solution (or another solution) on other devices. The notification is thus "pushed out" from one solution to multiple devices. The screenshot below shows how a PN is sent from a sending solution to receiving solutions, with the recipients in this case being identified on the basis of an external PN key (other methods for identifying recipient devices are also available).

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 **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 (i) with an **external PN key**, and/or (ii) 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 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.

### 11.5.1 The Sending Solution

In the sending solution, you select an event that will trigger the sending of the push notification (PN), and then define the **Send Push Notification** 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 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.

11.5.2 The Receiving Solution

This section lists the different parts of the Push Notification (PN) mechanism that are present in the receiving solution. While the mechanisms listed below are part of the receiving solution, note that the receiving solution can be the same solution as the sending solution. If this is the case, then the mechanisms of both the receiving and sending solutions are combined in one and the same solution.

Note: The Big Content of standard MobileTogether solutions is displayed only on Android and Windows devices. If you want Big Content to be displayed on iOS devices, then compile the receiving solution as an AppStore App.

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 by using one external PN key per solution. This key is a text string that is generated by the Register Ext PN-Key action (see screenshot below). If a PN is sent to a given external PN key, then all devices that are registered with that key will receive the PN.

The fact that one PN key can be registered per solution has two implications:

- The device can be registered with different external PN keys, but each PN key is tied to a specific receiving solution. If two solutions on the device use the same PN key, then a push notification that targets this key will be delivered to both solutions on that device.
- Since the same key can be generated by the same solution on other mobile devices, the external PN Key serves to identify a particular set of mobile devices. If a push notification is sent to a particular PN key, then devices that were registered with this PN key will receive the push notification.

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 definitions

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.

11.5.3 Push Notifications in AppStore Apps

AppStore Apps are MobileTogether apps that are compiled into different OS-based apps. These apps can be posted to the respective app stores for end users to download. (See the section AppStore Apps 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).

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 to which the PN is sent is correct.
- Check that the p8 file and the two keys (APNs and authentication key) have been uploaded.
- Check that the Bundle ID in the Firebase settings and in the generated AppStore App are identical.
- Check that PNs for your app ID in Apple’s Developer Console have been enabled.
- If none of the above apply, go to xcode/your project/capabilities. Switch off the PN switch, and then switch it on again.

Q: “Show as notification” does not work on iOS 9.
A: This is not an error. Support is provided for iOS 10 or later.

Windows

When generating program code for a Windows app of a PN-receiving solution, you will need, for Screen 6 of the Generate Program Code Wizard, two pieces of information: (i) the Package SID, and (ii) the Application Secret. The procedure to obtain these is as follows:

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 above).
5. The Registration page of the app will open. Note down the following details from this page; you will need them to generate the program code with the Generate Program Code Wizard: (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.
11.5.4 Simulating Push Notifications

A push notification (PN) contains data related to: (i) the PN's short message, (ii) the PN's big message, and (iii) the PN's payload. If the sending and receiving solutions are one and the same solution, then, in a simulation, the data transfer between sending and receiving parts is carried out within that one solution's simulation; the simulation in this case is straightforward.

However, if the sending solution and receiving solution are different, then the simulation mechanism is as follows: PN data sent during a simulation of the sending solution is recorded in a MT PN Simulation file (which has a .mtpnsim extension). When the receiving solution is simulated, you can load that .mtpnsim file. The simulator will now display all the sets of PN data in the .mtpnsim file, and you can select the PN that you want to simulate.

Recording PN Simulation Data

If, while simulating a sending solution, you trigger an event that sends a PN, then the dialog shown below appears.

![Record PN Data Dialogue]

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 from 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.

<table>
<thead>
<tr>
<th>Chart type</th>
<th>X-Axis (Category)</th>
<th>Y-Axis (Value)</th>
<th>Number of Series (on Z-Axis)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pie Charts (2D, 3D)</td>
<td>Text</td>
<td>Numeric</td>
<td>1</td>
</tr>
<tr>
<td>Bar Charts Ungrouped (2D, 3D)</td>
<td>Text</td>
<td>Numeric</td>
<td>1</td>
</tr>
<tr>
<td>Bar Charts Grouped (2D, 3D)</td>
<td>Text</td>
<td>Numeric</td>
<td>&gt; 1</td>
</tr>
<tr>
<td>Category Line Graphs</td>
<td>Text</td>
<td>Numeric</td>
<td>1 line = 1 series</td>
</tr>
<tr>
<td>Value Line Graphs</td>
<td>Numeric</td>
<td>Numeric</td>
<td>1 line = 1 series</td>
</tr>
<tr>
<td>Area and Stacked Area Charts</td>
<td>Text</td>
<td>Numeric</td>
<td>1 area = 1 series</td>
</tr>
<tr>
<td>Candlestick Charts</td>
<td>Text</td>
<td>Numeric</td>
<td>3 or 4</td>
</tr>
<tr>
<td>Gauge</td>
<td>—</td>
<td>Numeric</td>
<td>1</td>
</tr>
<tr>
<td>Overlay Charts</td>
<td>Text</td>
<td>Numeric</td>
<td>= 1 or &gt; 1 per chart</td>
</tr>
</tbody>
</table>

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

11.6.1 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 chart’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 **Chart Creation Settings**. The Chart Configuration dialog (screenshot above) of that chart appears. You can edit the chart's settings or data selection in the dialog, then click **OK** to finish.

11.6.2  Chart Data Selection

This topic contains simple examples to illustrate how chart data selection works.

XML file used in chart examples: YearlySales.xml

```xml
<?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:

<table>
<thead>
<tr>
<th>XPath</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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.

11.6.2.1 Chart Data Selection: Simple

This section:

- Introduction
- The context node
- Data selection for the X and Y axes
- If the For-Each expression returns items that are not nodes

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
<?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.
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:

![Yearly Sales: Regional Comparison](chart.jpg)

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:

<table>
<thead>
<tr>
<th>For-Each XPath</th>
<th>X-Axis</th>
<th>Y-Axis for Series</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Americas</td>
</tr>
<tr>
<td>2005</td>
<td>2005</td>
<td>XPath-1</td>
</tr>
<tr>
<td>2006</td>
<td>2006</td>
<td>XPath-1</td>
</tr>
<tr>
<td>2007</td>
<td>2007</td>
<td>XPath-1</td>
</tr>
<tr>
<td>2008</td>
<td>2008</td>
<td>XPath-1</td>
</tr>
<tr>
<td>2009</td>
<td>2009</td>
<td>XPath-1</td>
</tr>
<tr>
<td>2010</td>
<td>2010</td>
<td>XPath-1</td>
</tr>
</tbody>
</table>
11.6.2.2 Chart Data Selection: Flexible

This section:

- About flexible chart data selection
- One row, one series
- Three rows, three series, category values not merged
- Three rows, three series, category values merged
- One row, three series
- Rules for chart data selection

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
<?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:

![Yearly Sales: Regional Comparison](image)

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:

**Yearly Sales: Regional Comparison**

![Yearly Sales Chart](image)

**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.

11.6.2.3 Overlay Charts
Overlay charts (screenshot below) enable you to combine multiple charts of different types. The overlay chart in the screenshot below is explained in the text of the screenshot. The following types of chart can be combined:

- 2D bar charts and stacked bar charts
- Line charts
- Area charts and stacked area charts
- Candlestick charts
Yearly Sales by Region (Units Sold)

Chart type: Overlay Chart

The chart below has three layers. The bottom layer is an area chart showing the total sales of all three regions (Americas, Europe, Asia). The next layer is a bar chart layer which shows the sales of the region selected in the combo box. The third and topmost layer is a layer with three line graphs, showing the sales of each of the three regions separately. The three Y-Axes, starting from left, correspond to the three chart layers starting from the bottom layer.

Region to be displayed in bar chart: Americas

Creating the overlay chart
An overlay chart is created by selecting the Multiple Layers radio button in the Chart Configuration dialog (see screenshot below). A main layer is automatically created. You can select the type of chart for the main layer and then make the chart settings and data selection in the usual way.
Managing the layers of the overlay chart

You can add new layers to an overlay chart and delete layers. To do this, click the Manage button in the Chart Configuration dialog (see screenshot above). This displays the Overlays dialog (screenshot below). Click Add to add a new layer. To delete a layer, select a layer and click Delete.

You should note that when a layer is placed over another, it will obscure the layers beneath it. Since only area charts can be made transparent, some layer arrangements might not be optimal. For example a bar chart that is layered over a line chart would obscure parts of the lines. You should keep this in mind when planning the layering order.

Data selection for the various charts

The data selection for each chart depends on the chart type and is done in the usual ways for each chart type. See the sections, Chart Data Selection: Simple 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.

11.6.3 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).

11.6.3.1 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 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.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{sales_stack.png}
\caption{Yearly Sales: Regional and Total}
\end{figure}

\textbf{Line charts}

The difference between a line chart (\textit{below left}) and a value line chart (\textit{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.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{sales_line.png}
\caption{Yearly Sales: Regional Comparison}
\end{figure}
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**

A gauge chart is a type of chart that is used to visually represent a single value. It is characterized by a needle or pointer that moves in response to changes in the value being measured. Gauge charts are often used to display data in a way that is easy to interpret at a glance, making them a popular choice for dashboards and performance monitors.
Gauge charts are used to illustrate a single value and show its relation to a minimum and a maximum value.

Other basic settings
In the Chart Settings pane, you can also set the title of the chart (see screenshot below).
11.6.3.2 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.

### 11.6.3.2.1 General

The General section of the Change Appearance dialog box lets you define the title of the chart, add or remove a legend, and define background pictures and colors and—for bar, line, area, and candlestick charts—orientation of the chart.

#### Chart

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.
11.6.3.2.2 Type-Related Features

For each of the chart types, and even for the various sub-types, the Change Appearance dialog box provides a section where you can define the type-related features of the chart.

Pie chart

Most settings are the same for the 2d and 3d versions. In 2d pie charts, you can additionally draw highlights.

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 (this option is available for 2d bar charts only).
- The values of a bar (corresponding to the height of the bar on the Y-axis) can be drawn on the bar. The font of the values can be specified in the Fonts settings (this option is available only for 2d bar charts, not stacked).
- The distance between the series of a bar-group and between bar-groups can be specified as a decimal fraction of the width of a single bar. For example, in the screenshot below, which shows bar-groups that each consist of a blue series and a green series, the distance between the series has been set to 25% (=0.25) of the width of a bar; the distance between bar-groups has been set to 100% (=1.0) of the width of a bar. This option is available only for 2d bar charts.

![Bar Chart Screenshot](image)
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 than 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.

11.6.3.2.3 Colors

Depending on the chart type you have selected, MobileTogether Designer provides two different sections for the definition of colors to be used in charts:

- Color Schema for pie, bar, line, area, and candlestick charts
- Color Range for gauge charts
Color Schema

The Color Schema section of the Change Appearance dialog box provides four predefined color schemas (i.e., default, grayscale, colorful, and pastel) that can be customized; in addition you can also define your own color schema from scratch.

The top color will be used for the first series, then the second color and so on. You can change the order of the colors by selecting a color and dragging it to its new position with the mouse. To add a new or delete an unwanted color, click the corresponding button. In candlestick charts, only the first color will be used.

If you have appended one or several layers of overlay charts to a Charts window, the Color Schema section of the Change Appearance dialog box contains the additional radio button 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.

<table>
<thead>
<tr>
<th>Starting from</th>
<th>Fill with color</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

The definition shown in the screenshot above will appear in the gauge charts as follows:

11.6.3.2.4 X-Axis

In the X-axis section of the Change Appearance dialog box, you can enter a label for the axis, and define colors for the axis and the grid lines (if displayed). You can also define whether or not you want to display tick marks and axis values. This section is the same for all bar, line, area, and candlestick charts. The Show Categories options enables you to specify that only a subset of all categories (X-Axis values) are displayed, that is, only the ticks, grid lines, and values of the selected categories will be displayed. Create the subset of displayed categories by entering (i) the index of the first value to display, and (ii) the number of indices to step. For example, if there are 101 categories, from 1900, 1901, 1902 ... 1999, 2000, then you can show every tenth year from 1900 to 2000 by setting First index to 1 and Step to 10.
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 X-Axis. For example, if the values run from the 0 to 360, selecting this option will generate the X-Axis so that 360 is at the origin and the values progress down to 0 as the X-Axis goes upwards.

**Line**
The axis is displayed in the color that you choose from the Line drop-down-list. You can use one of the preselected colors, or click the Other color... button to choose a standard color or define a custom color. Click the Select... button on the Custom tab and use the pipette to pick a color that is displayed somewhere on your screen.

**Grid lines**
If the Show Grid lines check box is activated, you can choose a color from the corresponding drop-down list box.
Tick Interval
If you are not satisfied with the default tick marks, you can activate the Manual radio button in the Tick Interval group box and enter the difference between the individual tick marks into the corresponding field.

Tick Drawing
You can switch the display of tick marks on the axis and/or axis values on or off.

Axis Position
From the drop-down list, you can choose the position where the axis is to be displayed. When selecting "At Value / On Category Number", you can also position the axis anywhere within the plot.

11.6.3.2.5 Y-Axis
In the Y-axis section of the Change Appearance dialog box, you can enter a label for the axis, define colors for the axis and the grid lines (if displayed), define the value range, and decide if and where tick marks should be displayed and whether or not you want to show the axis values. This section is the same for all bar and line charts.

Label
The text entered into the Label field will be printed to the left of the axis as a description of the Y-axis.
Range
By default, the Auto radio button is selected in the Range group box. If you want to display a fragment of the chart in greater detail, activate the Manual radio button and enter minimum and maximum values into the respective fields. If the column that is used for the Y-axis does not include zero, you can deactivate the Include Zero check box and the Y-axis will start with the minimum value that is available in the series. The Invert Axis option enables you to invert the values of the Y-Axis. For example, if the values run from the 0 to 360, selecting this option will generate the Y-Axis so that 360 is at the origin and the values progress down to 0 as the Y-Axis goes upwards.

Line
The axis is displayed in the color that you choose from the Line drop-down-list. You can use one of the preselected colors, or click the Other color... button to choose a standard color or define a custom color. Click the Select... button on the Custom tab and use the pipette to pick a color that is displayed somewhere on your screen.

Grid lines
If the Show Grid lines check box is activated, you can choose a color from the corresponding drop-down list box.

Tick Interval
If you are not satisfied with the default tick marks, you can activate the Manual radio button in the Tick Interval group box and enter the difference between the individual tick marks into the corresponding field.

Tick Drawing
You can switch the display of tick marks on the axis and/or axis values on or off.

Axis Position
From the drop-down list, you can choose the position where the axis is to be displayed. When selecting "At Value / On Category Number", you can also position the axis anywhere within the plot.

11.6.3.2.6 Z-Axis
In the Z-axis section of the Change Appearance dialog box, you can enter a label for the axis, define colors for the axis, and decide whether or not you want to show tick marks on the axis. This section is the same for all 3d bar charts (Bar Chart 3d and Bar Chart 3d Grouped).
**Label**

The text entered into the Label field will be printed to the right of the axis as a description of the Z-axis.

**Line**

The axis is displayed in the color that you choose from the Line drop-down-list. You can use one of the preselected colors, or click the Other color... button to choose a standard color or define a custom color. Click the Select... button on the Custom tab and use the pipette to pick a color that is displayed somewhere on your screen.

**Tick Drawing**

You can switch the display of tick marks on the axis on or off.

### 11.6.3.2.7 3D Angles

In 3d bar charts you can customize the 3d appearance of the chart in the 3d Angles section of the Change Appearance dialog box.

![Camera](image)

The Field of view option causes the diagram to appear as if observed from a small or great distance. Values ranging from 1 through 120 are valid. Higher values cause the diagram to appear as if observed from a greater distance.

The Tilt value determines the rotation around the X-axis, whereas the Rotation value defines the rotation around the Y-axis. You can automatically adapt the size of the chart axis to the Chart window width by selecting the corresponding check box.

If the Automatic Chart Axis Size check box is selected, MobileTogether Designer will automatically calculate the optimum size of the X-axis as well as the Y-axis for the current Chart window size. The width and height of the chart will change dynamically when you resize the Chart window.

### 11.6.3.2.8 Sizes

In the Sizes section of the Change Appearance dialog box, you can define different margins as well as the size of axis and gauge ticks. Note that not all the properties listed below are available for all chart types.

**General**

- **Outside margin**  Space between the plot and the edge of the Chart window.
<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title to Plot</td>
<td>Space between the chart title and the upper edge of the plot.</td>
</tr>
<tr>
<td>Legend to Plot</td>
<td>Space between the lower edge of the plot and the legend.</td>
</tr>
<tr>
<td><strong>Pie</strong></td>
<td></td>
</tr>
<tr>
<td>Plot to Label</td>
<td>In pie charts, the space between the most left and right edge of the pie and its labels.</td>
</tr>
<tr>
<td>Pie Height</td>
<td>In 3d pie charts, the height of the pie.</td>
</tr>
<tr>
<td>Pie Drop Shadow</td>
<td>In 3d pie charts, the length of the shadow (if it is activated in the Pie section).</td>
</tr>
<tr>
<td><strong>X-Axis</strong></td>
<td></td>
</tr>
<tr>
<td>X-Axis to Axis Label</td>
<td>In bar and line charts, the space between the X-axis and its label.</td>
</tr>
<tr>
<td>X-Axis to Plot</td>
<td>In 2d bar charts and line charts, the space between the X-axis and the plot.</td>
</tr>
<tr>
<td>X-Axis Tick Size</td>
<td>In bar and line charts, the length of the ticks on the X-axis.</td>
</tr>
<tr>
<td><strong>Y-Axis</strong></td>
<td></td>
</tr>
<tr>
<td>Y-Axis to Axis Label</td>
<td>In bar and line charts, the space between the Y-axis and its label.</td>
</tr>
<tr>
<td>Y-Axis to Plot</td>
<td>In 2d bar and line charts, the space between the Y-axis and the plot.</td>
</tr>
<tr>
<td>Y-Axis Tick Size</td>
<td>In bar and line charts, the length of the ticks on the Y-axis.</td>
</tr>
<tr>
<td><strong>Z-Axis</strong></td>
<td></td>
</tr>
<tr>
<td>Z-Axis to Axis Label</td>
<td>In 3d bar charts, the space between the Z-axis and its label.</td>
</tr>
<tr>
<td>Z-Axis Tick Size</td>
<td>In 3d bar charts, the length of the ticks on the Z-axis.</td>
</tr>
<tr>
<td><strong>Line Drawing</strong></td>
<td></td>
</tr>
<tr>
<td>Connection Shape</td>
<td>In line charts, the size of the squares that mark the values in the chart.</td>
</tr>
<tr>
<td>Size</td>
<td>In line charts, the width of the line.</td>
</tr>
<tr>
<td><strong>3d Axis Sizes</strong></td>
<td></td>
</tr>
<tr>
<td>Manual X-Axis Size of Base</td>
<td>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.</td>
</tr>
<tr>
<td>Manual Y-Axis Size of Base</td>
<td>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.</td>
</tr>
<tr>
<td>Z-Axis Series Margin</td>
<td>In 3d bar charts, the distance on the Z-axis between the individual series.</td>
</tr>
<tr>
<td><strong>Gauge</strong></td>
<td></td>
</tr>
<tr>
<td>Border Width</td>
<td>In round gauge charts, the width of the border around the gauge.</td>
</tr>
<tr>
<td><strong>Gauge Ticks</strong></td>
<td></td>
</tr>
<tr>
<td>Border to Tick Distance</td>
<td>In round gauge charts, the space between the inner edge of the border and the ticks that mark the values.</td>
</tr>
<tr>
<td>Major Tick Length</td>
<td>In round gauge charts, the length of the major ticks (i.e., ticks that show a label).</td>
</tr>
<tr>
<td>Major Tick Width</td>
<td>In round gauge charts, the width of the major ticks (i.e., ticks that show a label).</td>
</tr>
<tr>
<td>Minor Tick Length</td>
<td>In round gauge charts, the length of ticks that do not have a value displayed.</td>
</tr>
<tr>
<td>Minor Tick Width</td>
<td>In round gauge charts, the width of ticks that do not have a value displayed.</td>
</tr>
</tbody>
</table>
Needle Length
In round gauge charts, the length of the needle. (Note that the percentage is calculated from the diameter of the gauge, so if you choose a value greater than 50%, the needle will point to somewhere outside the gauge!)

Needle Width at Base
In round gauge charts, the width of the needle at the center of the gauge.

Needle Base Radius
In round gauge charts, the radius of the base that covers the center of the gauge.

Gauge Color Range
Border to Color Range Distance
In round gauge charts, the space between the inner edge of the border and the outer edge of the color range.

Color Range Width
In round gauge charts, the width of the customizable color range. (Note that the percentage is calculated from the diameter of the gauge!)

Gauge Value Offset to Center
Distance from the center at which the gauge value is displayed.

Extra Value Offset to Center
Distance from the center at which the extra label (defined in the Gauge Chart settings) is displayed.

11.6.3.2.9 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 pieces of a pie chart
- **Axis Title**: The name of the X, Y, and Z axis in a bar or line chart
- **Axis Values**: The units displayed on an axis in a bar or line chart
- **Tick Values**: The units displayed on a gauge chart
- **Values**: The values displayed on the bars of a bar chart

### 11.6.3.3 Dynamic XPath Settings

Dynamic XPath Settings are very useful if you wish to use dynamic data from the XML document in the settings of the chart. For example, the title of a chart about a Region element might need to have data about that Region element (such as its name) in the title of the chart. If there are several Region elements, any one of which will be used at a time for the chart, then the data for the chart title can only be obtained dynamically via an XPath expression. The Dynamic XPath Settings dialog enables such data to be accessed via XPath expressions.

To access the chart's Dynamic XPath Settings dialog, click **Dynamic XPath Settings** in the Chart Settings pane of the Chart Configuration dialog (see screenshot below).

![Dynamic XPath Settings](screenshot)

**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'.

<table>
<thead>
<tr>
<th>Property</th>
<th>Value (XPath)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General Settings</strong></td>
<td></td>
</tr>
<tr>
<td>Background color</td>
<td></td>
</tr>
<tr>
<td>Background gradient end color</td>
<td></td>
</tr>
<tr>
<td>Background Fill Mode</td>
<td></td>
</tr>
<tr>
<td>Background image file</td>
<td></td>
</tr>
<tr>
<td>Background image mode</td>
<td></td>
</tr>
<tr>
<td>Plot background color</td>
<td></td>
</tr>
<tr>
<td>Plot Background gradient end color</td>
<td></td>
</tr>
<tr>
<td>Background Fill Mode</td>
<td></td>
</tr>
<tr>
<td>Plot Background image file</td>
<td></td>
</tr>
<tr>
<td>Plot Background image mode</td>
<td></td>
</tr>
<tr>
<td>Chart Title</td>
<td>concat(id, ': Yearly Sales (in units)')</td>
</tr>
<tr>
<td>Show legend</td>
<td></td>
</tr>
<tr>
<td><strong>Title Font</strong></td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td>'#023d7d'</td>
</tr>
<tr>
<td>Name</td>
<td>'Tahoma'</td>
</tr>
<tr>
<td>Bold</td>
<td>'1'</td>
</tr>
<tr>
<td>Italic</td>
<td></td>
</tr>
<tr>
<td>Underline</td>
<td>'yes'</td>
</tr>
<tr>
<td>Minimum fontsize</td>
<td></td>
</tr>
<tr>
<td>Maximum fontsize</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>'18pt'</td>
</tr>
<tr>
<td><strong>Legend Font</strong></td>
<td></td>
</tr>
<tr>
<td>Color</td>
<td></td>
</tr>
</tbody>
</table>

OK  Cancel
11.7 Control Templates

A control template is a template that you can reuse at multiple locations in the pages of your design. The main steps and key mechanisms for using control templates are listed briefly below and described in detail in the sub-sections of this section. The Example Projects section shows, through examples, the different ways in which control templates can be used.

Creating a control template

A control template is designed like a page. You can add controls to it that structure the design of the page and use data from page sources.

Note the following key features:

- You can create multiple control templates in a project.
- You can reuse a control template at different locations in the design.
- You can change the design of a page by switching templates.
- You can select a template based on runtime conditions or user input.
- You can modify the contents of a template based on runtime conditions or user input.

These uses are demonstrated in the project files described in the Example Projects section.

Using a control template via the Placeholder Control

After a control template has been created, you can use it via a Placeholder Control. This is a control which you (i) place at the location where you want to use the control template, and (ii) associate with the control template you want to instantiate there.

The Placeholder Control's properties specify the following:

- Which control template to use at the Placeholder Control's location.
- The XPath expressions that define the parameter values to pass to the selected control template.
- A new context node for XPath expressions that are used in the control template.

All of these properties are specified as values of the Placeholder Control's properties in the Styles & Properties Pane.

This section

This section consists of the following sub-sections:

- Creating a Control Template
- Using a Control Template: Placeholder Controls
- Overriding Template Events
- Example Projects
11.7.1  Creating a Control Template

Create a control template as follows:

1. In the Pages Pane, click the Add Page icon and select Add Control Template (see screenshot below).

2. Double-click the new control template item that is added to the list (see screenshot below) and rename it suitably. The next step will be to declare parameters and define variables for the control template.

Parameters and variables

Parameters enable you to use dynamic or static values within the scope of the control template when the template is instantiated. Both, parameters and variables, are declared in the control template. The main difference between the two is that while the values of variables are defined in the control template itself, the values of parameters are defined on the calling page and are passed to the control template's parameters when the control template is instantiated.

If you want to declare parameters and/or variables in a control template, in the Pages Pane (see screenshot above), click the template's Additional Options button (which is located to the right of the control template name). In the Parameters dialog that appears (screenshot below), add a parameter or variable (by clicking the Add Parameter or Add Variable toolbar icon in the top left of the respective pane) and name the parameter/variable suitably (by double-clicking and editing the name).
Parameters
Check a parameter's Optional box if you want to make the parameter non-mandatory. If a parameter is mandatory, it is an error if it does not receive a value at run time. If, in the calling placeholder control, the value of a parameter is defined by an XPath expression, then this expression is evaluated in the context of the placeholder control's context. The context node is not changed by the Control XPath Context property of the placeholder control that instantiates the control template.

Variables
To enter the value of a variable (dynamic or static), double-click in the variable's Value field and enter a suitable XPath expression.

Note the following points about how the XPath expressions of parameters and variables are evaluated:

- The context of XPath expressions will be the context node set in the Control XPath Context property of the calling placeholder control. If no value has been set for this property, then the context will be the context node of the calling placeholder control.
- This XPath expression to generate variable values can use the parameters of that control template as well as variables defined earlier in the list of variable definitions. For example: If a control template has parameters $a, $b, $c, and variables $x, $y, $z (in that order), then the variable $y can use this list of parameters and variables to generate its value: $a, $b, $c, $x, (but not $z).

After you have finished declaring parameters and variables for the control template, click OK. The parameters you declared are displayed in the Pages Pane as shown for the DataWithParams template in the screenshot below. You can edit the parameter/variable list of a control template at any time by clicking the control template's Additional Options button to open the Parameters dialog.
Note: You can change the order of parameters and variables by dragging a parameter/variable and dropping it to a new location.

Note: If you change the order of parameters, you need to make sure that the parameter order is correct in the call to the control template (which is made via placeholder controls). Do this as follows: (i) Right-click the control template and select List Usages; (ii) For each usage (the list is shown in the Listings Pane), check whether the control template is selected via the dropdown list or via an XPath expression; (iii) If the control template is selected via an XPath expression and if its parameters are selected via an array expression (rather than a map expression), then make sure that the order of the parameters in the array is the same as that in the parameter definitions list of the Parameters dialog (see screenshot further above).

Content of the control template
To display the control template's design canvas, select the control template in the Pages Pane (see screenshot above). The design canvas will be displayed in Design View. Here, you can set up and design a set of controls as you do for a normal page. You can use controls, actions, page source nodes, and other components as usual.

Note the following points:

- Only the first visible control in sequential order will be displayed when the control template is instantiated. (A visible control is one that has its Visible property set to true.) This enables you to create multiple controls inside the template and to determine which of these is used by setting XPath conditional expressions on the Visible property of the respective controls. The first control with a Visible property that evaluates to true will be used.
- If you wish to display multiple controls when the control template is instantiated, add these controls inside the cells of a Table control with repeating rows or columns—since the table counts as a single control. See screenshot below, where a label has been added to each cell of a single row of the table, and the row itself can repeat. Tables with repeating columns and rows are also used in the example projects. Note, however, that you cannot add repeating tables (in which the entire table repeats).
- If you need to use a node from one of the design's page sources and the page source is not shown in the Page Sources Pane, then you can add the page source via the pane's Add Source icon.
In the control template's design, you can also use the parameters and variables that you declared for the control template. For example, in the `DataWithParams` control template shown in the screenshot below, a `Table control` has been created with a single row of seven cells, each of which has a label. Each label has a `Text` value that is generated by an XPath expression that selects a parameter or a variable. The parameter values will be supplied by the calling page and will become the texts of the labels. This control template can be used at different locations and can have different label texts according to the parameter values passed to the template from the calling page.

You can style the controls in a control template in the same way that these controls are usually styled (via `their properties` and via `stylesheets`).

After the control template has been created as described above, it can be used in `top pages and sub pages`, as well as in control templates themselves.

See `Example Projects` for examples of how to use control templates.

### The context node of control templates

The context node of a control template will be the context node of the `Placeholder Control that calls the control template`. If you want to change the context node, you can specify a new context node via the `Control XPath Context` property of the calling placeholder control. You must be aware of what the context node is, since all XPath expressions in the control template must work correctly within this context. If you plan to reuse the control template at various design locations, make sure that the XPath expressions in the control template are correct in each of these different contexts.

### Editing parameters and variables of control templates

You can edit the parameters and variables of a control template (add new, delete selected, or modify properties of) at any time. To do this, go to the control template's Parameters dialog by clicking the `Additional Options` button of the control template. This button is located to the right of the control template name in the `Pages Pane` (see screenshot below).
11.7.2 Using a Control Template: Placeholder Controls

After you have created a control template, you can use it at one or more locations in page-type components: (i) Top Pages, (ii) Sub Pages, (iii) Control Templates.

To use a control template in a page or control template, do the following:

1. Drop a Placeholder Control in the design, either directly on the page or inside a suitable control such as the Table control.
2. Set the Placeholder's Control Template property to select the control template you want to instantiate at this location.
3. Configure the other settings of the Placeholder Control as required in the Styles & Properties Pane.

Inserting a Placeholder Control that calls a control template in a table with repeating elements is especially useful since it enables the template to be applied to each repeating element of the table. In the example shown in the screenshot below, for instance, we have created a table with repeating rows where each of the repeating rows corresponds to a country in an XML dataset that contains multiple sibling Country elements. Then we dropped a Placeholder Control into the table's repeating row and set its Control Template property to select the DataWithParams template.

As you can see in the screenshot, the control template has seven cells for seven different country properties. In the current placeholder, three of the properties are instantiated. In another placeholder a different set of properties could be instantiated, enabling the same control template to be called at different locations to generate different sets of data.

Also see the Example Projects section for more usage examples.

Configuring the Placeholder Control

Configure a Placeholder Control as follows:

1. After you have dropped the Placeholder Control in the design, select it in the design.
2. Go to the Styles & Properties Pane and make sure that the Placeholder Control's Control Template property is correctly set to the name of the control template that you want to insert. (In the screenshot below, the selected control template is DataWithParams; it is selected from a dropdown that lists all the control templates in the design.) Alternatively, you can use an XPath expression to select a control template. Such an XPath expression must evaluate to the name of a control template in the design. Using XPath expressions enables you to select templates conditionally. See Example Projects for ways to do this.
3. After you select the control template, its parameters are displayed as sub-properties of the template (see screenshot below). Enter values for these parameters as XPath expressions. Values must be defined for all mandatory parameters (those that have not been defined as optional). The context node for evaluating XPath expressions of parameter values is the context node of the Placeholder Control. This context for parameters is not changed by the Control XPath Context property (see below).

If you select a control template via an XPath expression (rather than by selecting a name in the property's combo box), then the control template is not selected till runtime, when the expression is evaluated. Since the control template is not known at design time, no parameters can be displayed in the pane. Instead of a list of parameters, a property named Template Parameters is available (see screenshot below). You can enter an XPath expression to generate the values of the expected parameters. The expression must be either an array expression or map expression. If you use an array expression, the parameter values must be provided in the same sequence as the parameter definition order in the control template; additionally any optional parameters must not be omitted. (In the case of maps, the keys enable the values to be correctly assigned.) See Example Projects for a sample of such expressions.
4. The context node of all XPath expressions inside a control template will by default be the context node of the placeholder control that instantiates (or calls) the template. You can change the context node that is passed to the control template by entering a new context node as the value of the Control XPath Context property (see screenshot above). The new context node will be used to evaluate XPath expressions within the template, including expressions that define the values of template variables.

11.7.3 Overriding Template Events

This is an advanced feature that enables you to specify, on each Placeholder Control, a set of actions to execute when the placeholder's control template is executed. This set of actions overrides the actions defined in the control template, and is triggered when any action event defined in the control template occurs. It is therefore possible to define a different set of overriding actions for each placeholder control. In this way a control template can be modified at the placeholder level, separately for each placeholder control. For example, if three placeholder controls use a single control template, and you want to modify the control template when it is instantiated by one of the three placeholder controls, then you can define a set of overriding actions on this particular placeholder control.

How to override control-template actions

On a Placeholder Control, you can define a set of actions that override the control actions defined in a control template. Do this as follows:

1. Select the Placeholder Control for which you want to modify control-template actions.
2. In the Styles & Properties Pane, go to the placeholder control's properties and click the Additional Options button of its Control Action property (screenshot below). Alternatively, right-click the placeholder control and, in the context menu that appears, select Template Control Event Overrides.
3. In the Actions window that appears, enter the set of actions you want in the Template Control Event pane (see screenshot below).

In the screenshot above, the placeholder control's actions have been modified by means of an If-Then-Else action, as follows:

- If the language of the client device is English, then the control template is executed as usual. This is achieved by adding, to the Then branch, a Template Event Callback action (shown highlighted in the Actions pane at left in the screenshot above). This action simply instantiates the control template.
If the client language is not English, then a message box is displayed. The Message Box action is added to the Else branch.

In the event that the control template contains more than one control, you can select different actions for each control event by using conditional processing that tests the value of the $MT_ControlKind or $MT_ControlName variables.

### 11.7.4 Example Projects

There are four example projects in this section, each of which builds on the previous project. The first project shows how to create a control template and use it. The goal is a simple one: let the user select a continent, then a country, and a city (see screenshot below). A selection at each level generates a list of radio buttons at the next level. We reuse control templates to generate these radio button lists. Subsequent projects add functionality and demonstrate different ways in which control templates can be used.

To open and experiment with any of the example projects, simply double-click its MTD file (see below for location).

**Location of project files**

All the example project files listed below are located in the *(My) Documents* folder:

```
Altova\MobileTogetherDesigner7\MobileTogetherDesignerExamples\Tutorials\ControlTemplates
```

- Cities1-Reuse.mtd
- Cities2-SwitchTemplates.mtd
- Cities3-DynamicUpdates.mtd
- Cities4-DynamicSelection.mtd
- CitiesWorldwide.xml

**Listing of CitiesWorldwide.xml**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Cities in which the survey is being carried out -->
<Cities>
  <Continent name="Americas">
```
<Country name="USA">
  <City name="Miami"/>
  <City name="San Francisco"/>
  <City name="Chicago"/>
  <City name="Washington"/>
  <City name="Los Angeles"/>
  <City name="Boston"/>
  <City name="New Orleans"/>
  <City name="Denver"/>
  <City name="Seattle"/>
  <City name="New York"/>
  <City name="Philadelphia"/>
  <City name="Minneapolis"/>
</Country>

<Country name="Canada">
  <City name="Vancouver"/>
  <City name="Ottawa"/>
  <City name="Toronto"/>
  <City name="Montreal"/>
  <City name="Calgary"/>
  <City name="Edmonton"/>
</Country>

<Country name="Mexico">
  <City name="Mexico City"/>
  <City name="Guadalajara"/>
  <City name="Puebla"/>
  <City name="Tijuana"/>
  <City name="Monterrey"/>
</Country>

</Continent>

<Continent name="Europe">
  <Country name="UK">
    <City name="London"/>
    <City name="Edinburgh"/>
    <City name="Cambridge"/>
    <City name="Oxford"/>
    <City name="Liverpool"/>
    <City name="Bristol"/>
  </Country>

  <Country name="Germany">
    <City name="Heidelberg"/>
    <City name="Stuttgart"/>
    <City name="Berlin"/>
    <City name="Frankfurt"/>
    <City name="Hamburg"/>
    <City name="Bonn"/>
    <City name="Munich"/>
  </Country>

  <Country name="France">
    <City name="Paris"/>
    <City name="Marseille"/>
    <City name="Lyon"/>
    <City name="Toulouse"/>
    <City name="Nice"/>
    <City name="Strasbourg"/>
  </Country>
</Continent>
</Country>

<Country name="Italy">
   <City name="Rome"/>
   <City name="Milan"/>
   <City name="Naples"/>
   <City name="Turin"/>
   <City name="Palermo"/>
   <City name="Genoa"/>
   <City name="Bologna"/>
   <City name="Florence"/>
   <City name="Venice"/>
</Country>

</Country>
</Continent>

<Continent name="Asia">
   <Country name="Japan">
      <City name="Tokyo"/>
      <City name="Osaka"/>
      <City name="Nagoya"/>
      <City name="Sendai"/>
      <City name="Sapporo"/>
   </Country>
   <Country name="China">
      <City name="Beijing"/>
      <City name="Shanghai"/>
      <City name="Tianjin"/>
      <City name="Hong Kong"/>
   </Country>
   <Country name="India">
      <City name="New Delhi"/>
      <City name="Mumbai"/>
      <City name="Chennai"/>
      <City name="Bengaluru"/>
      <City name="Kolkata"/>
   </Country>
</Continent>

<Continent name="Oceania">
   <Country name="New Zealand">
      <City name="Auckland"/>
      <City name="Wellington"/>
      <City name="Christchurch"/>
   </Country>
   <Country name="Australia">
      <City name="Canberra"/>
      <City name="Melbourne"/>
      <City name="Brisbane"/>
      <City name="Sydney"/>
      <City name="Adelaide"/>
      <City name="Hobart"/>
   </Country>
</Continent>
In this section

This section consists of the following topics:

- Reusing a Template
- Switching Templates
- Dynamically Changing Template Content
- Dynamically Selecting a Template

11.7.4.1 Reusing a Template

One of the main benefits of control templates is reuse. After you have created a control template, you can use it in different locations in your design. In our example Cities1-Reuse.mtd, the goal is to let users select their city by choosing continent, country, and city in that order as shown in the screenshot below. (All the geographic data is referenced from the XML page source CitiesWorldwide.xml.)
Since we are using two sets of horizontal radio buttons (one for countries and one for cities), we have created a single control template, named **Dynamic Radio Buttons (Horizontal)**, and used it twice: once to generate radio buttons of the countries of the selected continent, and a second time to generate radio buttons of the cities of the selected country.

**The project design**

The project consists of the control template **Dynamic Radio Buttons (Horizontal)**, which is used twice in the top page named **Select Country and City**. The design of the top page is shown in the screenshot below left. After the user selects a continent in the combo box, radio buttons of the countries in that continent are displayed. After a country is selected, radio buttons of the cities in that country are displayed.

The continent, country, and city selected by the user are each stored in separate nodes of the **$PERSISTENT** page source (see screenshot above right). Dropping these three page source nodes (the @Continent, @Country, and @City attributes of the **$PERSISTENT/Root** element) on their respective design components performs certain important functions:

- **@Continent** on the combo box: updates @Continent, with the selection in the combo box and makes @Continent the context node of the combo box.
- **@Country** on placeholder-control-1 makes @Country the context node of the placeholder control (and of the control template that the placeholder control calls). The value in @Country is updated via the OnFinishEditing action of the radio button in the control template (see below).
- **@City** on placeholder-control-2 makes @City the context node of the placeholder control (and of the control template that the placeholder control calls). The value in @City is updated via the OnFinishEditing action of the radio button in the control template (see below).

The setting of a context node for the placeholder control is important because all XPath expressions in the control template called by the placeholder control will be evaluated relative to this context node.
The control template "Dynamic Radio Buttons (Horizontal)"

The control template has been declared to have one mandatory parameter named $values. You can see this declaration by clicking the Additional Components button in the Pages Pane (circled in red in the screenshot below).

The design of the control template is shown in the screenshot below. The control template consists of a table that expands horizontally, that is, a new column is created for each item of the sequence over which the table expands. This sequence (for the expansion of the table) is defined to be that held in the $values parameter (as shown in the screenshot below). Each column contains a radio button. So, if $values holds a sequence of country names, then a column containing a radio button will be created for each country name in the sequence. And if $values holds a sequence of city names, then a radio button will be created for each city name in the sequence. How the contents of $values is defined is explained is explained below.

The radio button control has the following settings (defined in the Styles & Properties Pane):

- A Text property that determines the text that accompanies each radio button. The text in our example is defined to be the contents of the $MT_TableColumnContext variable. This variable is a MobileTogether application variable that has been specially devised for use with tables that expand horizontally. It contains, for each column of such tables, the current context node. So, in our example, if a new column is created due to a country name in the $values sequence, then that country name is the context node for that column and that country name will be stored in $MT_TableColumnContext variable while that column is being generated. Since the Text property of the radio button has been set to $MT_TableColumnContext, each radio button will have the name of the sequence item that caused the generation of the current column.

- The Control Action property (for OnFinishEditing) updates the current node with the value of $MT_TableColumnContext. So: firstly, what is the current node? It is the context node of the calling placeholder control (see above). In our example, it is, respectively (for the first and second placeholder), the @Country and @City attributes of the $PERSISTENT/Root element. Secondly, which $MT_TableColumnContext value (that is, of which column) will be passed to the current node? Answer: The column containing the radio button that was clicked. Consequent to the above: The country/city name of the clicked radio button will be passed to the @Country or @City attribute, respectively.
• The Checked Value (or Get Value from XPath) property of the radio button defines whether the button has been selected (a value of 1) or not (a value of 0). The XPath expression that determines this value in the control template is: `if ($MT_TableColumnContext = .) then 1 else 0`. This expression sets the value of that radio button to 1 that has a `$MT_TableColumnContext` value that is equal to the value of the current node (which is the context node of the placeholder control and, in our example, either the `@Country` or `@City` attribute of the `$PERSISTENT/Root` element). Since the current node was updated with the value of `$MT_TableColumnContext` via the `OnFinishEditing` action, only the radio button that was clicked (and triggered the `OnFinishEditing` action) will have a value of 1.

All that is needed now is to specify the sequence that is passed to the control template's `$values` parameter. This definition is set on the placeholder control (see below).

**Note:** The **Dynamic Radio Buttons (Horizontal)** enables you to create horizontal radio buttons for any sequence of values that is supplied to the `$values` parameter.

**The two Placeholder Controls that reuse the control template**

A **Placeholder Control** is created for a control template by dropping the Placeholder Control in the design at the required location, and setting its **Control Template** property to select the control template you want. Alternatively, you can drop the wanted control template in the design at the required location; the **Control Template** property will be set automatically.

The key settings for the Placeholder Control in our example are explained below. The screenshot above shows the properties of the first Placeholder Control, which is used to select countries.

- **Associated Page Source Node**: This specifies the page source node that will be the context node of the Placeholder Control, and, by extension, the context node of the control template that is called. All XPath expressions in the control template (including for its parameters and variables) will be evaluated relative to this node. You can make this setting by dropping the page source node onto the **Placeholder Control**. In our project the associated page source nodes for the two **Placeholder Controls** are, respectively, the `@Country` and `@City` attributes of the `$PERSISTENT/Root` element.
- **Control Template**: Selects the control template that will be instantiated at the location of the **Placeholder Control**. You can select the control template in the property's combo box. (Alternatively, you can drag a control template into the design and this property will be set automatically to select that control template.)
- **Parameters**: Defines the values to be sent to the `$values` parameter of the control template. In the case of the Placeholder Control that instantiates radio buttons for the countries of the user-selected
continent, the XPath expression is: \( \text{\$COUNTRIES-AND-CITIES/Cities/Continent[@name=\$PERSISTENT/Root/@Continent]/Country/@name} \). The predicate filter in this expression (highlighted in yellow) selects the continent that the user has chosen (and which is now stored in the \$PERSISTENT tree). The XPath expression of the second Placeholder Control, which generates radio buttons for cities of the selected country, is: \( \text{\$COUNTRIES-AND-CITIES/Cities/Continent[@name=\$PERSISTENT/Root/@Continent]/Country[@name=\$PERSISTENT/Root/@Country]/City/@name} \).

Notice that the two placeholders have different context nodes and send different sequences to the \$values parameter. This enables the control template to generate the appropriate radio buttons in the two locations where it is instantiated by the respective Placeholder Controls.

11.7.4.2 Switching Templates

A second benefit of control templates is the ability to easily change templates in the Placeholder Control, for example, if you want to change the layout of a component. In the example Cities1-Reuse.mtd (described in the previous topic), we run into the following problem: While the list of cities in some cases is short enough to fit in the horizontal span of the display (such as for Spain), some lists are too long to fit (for example, Germany); see screenshots below.

In order to fit all cities in the display, it would be more sensible to display cities in a vertical list. In the example Cities2-SwitchTemplates.mtd, we have added (to the previous design) a new control template that generates vertical radio buttons, and in the placeholder control that instantiates the radio buttons for cities, we have simply switched the control template to call the new control template. A simulation of the modified design is shown in the screenshot below.
The new control template: "Dynamic Radio Buttons (Vertical)"

The control template has been declared to have one mandatory parameter named $values (just like the control template for horizontal radio buttons). Additionally, however, a variable named $source has been defined with a value that is the current node.

The design of the control template is shown in the screenshot below. The major difference from the template for horizontal radio buttons is that the table does not expand horizontally (creating a new column for each item in $values). Instead, a new row is generated for each item in $values. Since each row contains a radio button, the radio button list is vertical.

The radio button control has the following settings (defined in the Styles & Properties Pane):

- A Text property that selects the current node. Since the current node inside a table row will be the current $values item for which the row is being generated, the text of each radio button will be the current $values item.
- The Control Action property (for OnFinishEditing) updates the $source variable with the value of the current node (which is the current $values item). As a result, the $source variable will contain the text value of the clicked radio button.
- The Checked Value (or Get Value from XPath) property of the radio button defines whether the button has been selected (a value of 1) or not (a value of 0). The XPath expression that determines this property value is: if ($source = .) then 1 else 0. This expression sets the value of that radio button to 1 that has a current node value which is equal to the value of $source. This will be true only
for the clicked radio button, since $source will be updated with the current node value only when the OnFinishEditing action is triggered (see previous point).

You now have a new control template that generates vertical radio buttons for the items in the template's $values parameter. Since this is the same parameter input that the Dynamic Radio Buttons (Horizontal) control template takes, you can now set a Placeholder Control to call either the template for horizontal radio buttons or that for vertical radio buttons.

Switching between templates in the Placeholder Control

In the example file, we go to the top page and, in the design, select the placeholder that instantiates the radio buttons for cities. In the Styles & Properties Pane, we can now switch between Dynamic Radio Buttons (Vertical) for vertically listed radio buttons and Dynamic Radio Buttons (Horizontal) for horizontal radio buttons (see screenshot below). In our example, we have chosen to display cities in a vertical list.

11.7.4.3 Dynamically Changing Template Content

Another benefit of using control templates is that you can pass parameter values to the control template and dynamically change content inside the template. In our example file Cities3-DynamicUpdates.mtd, we have added sort functionality to the previous example, Cities2-SwitchTemplates.mtd. It enables users to decide whether the display of countries and/or cities should be sorted (see screenshot below).
How it works
To implement the sort functionality, we did the following:

- Added an optional $sort parameter to both control templates.
- In the design of the top page, added a check box control on the right-hand side of each placeholder control (see screenshot below), and set the check box’s column-width to wrap_content. We also set the Visible property of each check box appropriately.
- In the $PERSISTENT tree, added two new attributes, SortCountries and SortCities, and set the default fixed value of each to 0 (see screenshot below). We then set these two nodes to be updated by the respective check-box value (selected or not-selected); this was done by dragging each node on to its respective check box.
- For each placeholder, defined the value of the $sort parameter to be the XPath expressions, respectively, if ($PERSISTENT/Root/@SortCountries=1) then true() else false() and if ($PERSISTENT/Root/@SortCities=1) then true() else false().
- In each control template, the sequence that generates the repeating rows or repeating columns of the table is defined by the XPath expression if ($sort) then sort($values) else $values. In plain language, this means: If the value of the $sort parameter is true(), then sort the items in the $values sequence, otherwise use the $values sequence as it is (which is the order in the XML data file CitiesWorldwide.xml).
What happens is this:

- The values of the `SortCountries` and `SortCities` attributes are set to an initial value of 0. The values of both can be changed by selecting (value set to 1) or deselecting (value set to 0) their respective check boxes.
- The values of the respective attribute nodes (`SortCountries` and `SortCities`) are passed via the two placeholders to the `$sort` parameter of the respective control template as either `true()` or `false()`.
- The XPath expression that selects the items of the `$values` sequence—which produces the repeating rows or repeating columns—either sorts the sequence or not, depending on the value of the template's `$sort` parameter (`true()` or `false()`).

### 11.7.4.4 Dynamically Selecting a Template

A powerful benefit of using control templates is that you can cause a template to be selected dynamically, depending on the runtime situation. For example, a different template can be selected according to user input or according to the device's environment or settings.

In our example file `Cities4-DynamicSelection.mtd`, a horizontal or vertical template is selected according to the orientation of the mobile device (landscape or portrait). If the device is held in landscape orientation, then horizontal radio buttons are displayed **automatically**. If the user changes the device's orientation to portrait, then the display of radio buttons is **automatically** changed to a vertical display. This design feature has been added to the [previous example](#), `Cities3-DynamicUpdates.mtd`.

**Note:** You can try out this example by turning the device (between landscape and portrait orientations) while the solution is running or by simulating a change of device orientation. To simulate a change of orientation while a simulation is running, click the **Change Orientation** toolbar button ([see screenshot below](#)).
The screenshot above shows a simulation of Cities4-DynamicSelection.mtd in landscape orientation. In it, the control template Dynamic Radio Buttons (Horizontal) has been automatically selected (because of the landscape orientation) for both sets of radio buttons, and, as a result, the device automatically displays radio buttons horizontally.

**Note:** The device orientation is known to us because it is stored in the two MobileTogether variables $MT_Landscape$ and $MT_Portrait$ (each of which have a value of either true() or false()).

**How it works**

In order to dynamically select templates, we need, in each Placeholder Control, to define two settings:

- an XPath expression to specify the conditions under which the alternative templates are respectively selected (the relevant property is highlighted in the Styles & Properties Pane in the screenshot below)
- an XPath expression to pass parameter values to the selected control template

**Selecting the control template**

The XPath expression for the placeholder that instantiates radio buttons for countries is:

```xml
if ($MT_Landscape)
then 'Dynamic Radio Buttons (Horizontal)'
else 'Dynamic Radio Buttons (Vertical)'
```
This expression specifies that if the $MT_Landscape global variable has a value of true(), then the control template named Dynamic Radio Buttons (Horizontal) must be used, otherwise (when $MT_Landscape=false()) the control template named Dynamic Radio Buttons (Vertical) must be used. Remember that this expression must evaluate to the name of the control template to be called, which it now does for each orientation (landscape and portrait).

The XPath expression for the placeholder that instantiates radio buttons for cities is slightly different. Since some countries display a large number of cities (Italy 9 and USA 12), all cities of such countries might not be able to be displayed horizontally in a single screen width, even in landscape orientation. So we have restricted the selection of the horizontal template (in landscape orientation) to only those countries having less than 9 cities. The XPath expression is listed below, with the part that counts the number of cities of the selected country being highlighted in yellow.

```xml
if ($MT_Landscape and count($COUNTRIES-AND-CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent]/Country[@name = $PERSISTENT/Root/@Country]/City/@name) lt 9)
then 'Dynamic Radio Buttons (Horizontal)'
else 'Dynamic Radio Buttons (Vertical)'
```

The effect of the definitions listed above is that: (i) in landscape orientation, all lists of countries are horizontal, whereas in portrait orientation they are all vertical; (ii) in landscape orientation, lists of 1 to 8 cities are displayed horizontally while lists containing more than 8 cities are displayed vertically; in portrait orientation all city lists are displayed vertical.

### Defining the parameter values to pass to templates

The XPath expression to pass values to the parameters of the selected control template (see screenshot below) must be either an array expression or a map expression. Bear in mind that you do not know which template will actually be called at runtime. However, you do know that the choice is between two templates and that they both happen to have two parameters with the exact same names. Because both parameters have the same name, the XPath expression to assign values is simpler.

For the first placeholder, we have used an array expression (listed below). The array must consist of two sequences (because we have two parameters), with the first sequence (highlighted in yellow below) providing the value of the first parameter and the second sequence (highlighted in blue) providing the value of the second parameter. The first sequence generates the value of the $values parameter and will evaluate to a list of countries of the selected continent. The second sequence generates the value of the $sort parameter and will evaluate to true() or false(). Note that the order of the sequences in the array must correspond to the order
of the parameter definitions in the project, and must include values for any optional parameters—which can be the empty sequence () if no value is needed.

```
[
    ($COUNTRIES-AND-CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent]/Country/@name),
    (if ($PERSISTENT/Root/@SortCountries=1) then true() else false())
]
```

For the second placeholder, we have used a map expression (listed below). The map must consist of key:value pairs, where the key must be the name of the parameter. The key:value pairs can be supplied in any order.

```
map{
    "values": {$COUNTRIES-AND-CITIES/Cities/Continent[@name=$PERSISTENT/Root/@Continent]/Country[@name = $PERSISTENT/Root/@Country]/City/@name},
    "sort": if ($PERSISTENT/Root/@SortCities=1) then true() else false()
}
```
11.8 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, copying, and deleting user-created style sheets

There are two types of style sheets: (i) a Project style sheet which is applied automatically at the project level and cannot be deleted; (ii) user-created style sheets, which can be applied separately to individual pages, tables, and controls. See Style Sheet Type and Scope for more information.
Delete Style Sheet

- To add a user-created style sheet, click **Add Style Sheet**.
- To copy a project or user-defined style sheet, do the following: (i) select the style sheet's name (in the screenshot above, for example, the name of one style sheet is *Project* and the name of the other is *Pastel*), (ii) press **Ctrl+C** to copy to the clipboard, (iii) press **Ctrl+V** to paste as a new style sheet. Alternatively you can use the **Copy** and **Paste** commands of the style sheet's context menu. This way you can start a new style sheet so that it already contains the styles of a pre-existing style sheet; you can then add or modify styles in the new style sheet.
- To rename a user-created style sheet, double-click the style sheet's name and edit it.
- To delete a user-created style sheet, click **Delete Style Sheet**.

**Defining styles**

In the left-hand pane, within a style sheet, select a level (page, table, or control) at which you wish to define a style, then, in the right hand pane, assign a value to that particular style property. You can select or enter a static property value, or you can enter an XPath expression that evaluates to a property value. An example of a dynamic assignment would be to make a property value conditional on some criterion, such as the screen width of the end-user mobile device.

**Priority of style definitions**

The closer the location of a style definition is to a component, the higher will be that style definition's priority (see *Priority across Style Sheets*) 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.

### 11.8.1 Style Sheet Type and Scope

On the basis of their scope, style sheets can be grouped into two types:

- A project style sheet named *Project*; this name is fixed and cannot be changed. The styles defined in the *Project* style sheet are **applied automatically** across the entire **project** (that is, across all the pages of the project). *Project* styles can be overridden by styles defined at a location with a higher priority.
- User-created style sheets, each with a user-given name. You can create any number of these style sheets. (In the screenshot above, there are two user-created style sheets: *Pastel* and *Taxi.*) The styles defined in a user-created style sheet can be applied to individual controls, tables, and/or pages. This is done by entering the name of the user-created style sheet as the value of the **Style Sheet** property of the respective control, table, or page. When a user-created style sheet is applied in this way, it will have a higher priority than the *Project* style sheet.

**Note:** To add a user-created style sheet, click **Add Style Sheet**. To rename a user-created style sheet, double-click the name and edit.
Note: If a style group (Controls, Table, or Page) contains at least one style definition, then that style group and its containing style sheet are displayed in bold; otherwise these are displayed in normal fontface. For example, in the screenshot above, no style has been defined in the Taxi style sheet, but at least one style has been defined in the other two style sheets. In the Project style sheet, at least one style has been defined for at least one control. The Pastel style sheet has at least one style defined in the Page group of styles and for at least one control.

Note: Each pane of the dialog (left and right) has an icon that enables/disables the display of non-empty items. Displaying only the non-empty items is useful when you wish to see a list of only the styles that have been defined; for example, when you want an overview of currently defined styles. The left-hand pane also has toolbar icons for (i) expanding all items, and (ii) collapsing all items.

11.8.2 Priority within a Style Sheet

Both the Project style sheet and user-created style sheets are structured into three levels:

```
Style Sheet (Level-1)
|-- All Controls (Level-2)
|   |-- ControlType-1 (Level-3)
|   |   . . .
|   |-- ControlType-n (Level-3)
|-- Table (Level-2)
|-- Page (Level-2)
```

You can also see this hierarchy in the screenshots below.
Each higher level in the hierarchy passes all its style properties to the level below it. So the All Controls level passes all its style definitions to the control types on Level 3. The principle is that, if a property value is set at the All Controls level, then all the controls (on the lower level) that have this property will inherit the property value that was set at the All Controls level.

The properties that are available at the Style Sheet level are all the properties of the All Controls level—plus the Table and Page properties (which define, respectively, properties for tables and pages). So, for example, if you set the Background Color property on the Style Sheet level (say a value of red, as in the screenshot below left), then all the control types of Level 3 that have a Background Color property, plus any tables in the design, as well as the design page itself will all inherit this value (red in this case)—as long as no definition exists for these descendant properties.

If you wish to override a property value that was assigned at a higher level, then assign an overriding value at the lower level. In the screenshot below right, for example, the Button control type has been assigned a Background Color property value of blue. So while all control types that have a Background Color property (as well as tables and the page) will inherit a background color of red (from the higher-level assignment in the screenshot at left), all Button controls will have a background color of blue. If you wish to give one specific Button instance a background color other than blue, specify the color you want in that particular Button instance's Background Color property. (Do this by selecting the Button control in the design and setting its Background Color property value in the Styles and Properties Pane.)
Higher priority for definitions located closer to the design component

If a style property exists at multiple levels, then the definition that is relatively more specific to the design component has relatively higher priority. For example, a style sheet property definition on a control type has higher priority than a definition for the same property on the style sheet level.

The table below gives, for each column, the relative priority levels of the same style property if the property is set at multiple levels. Levels lower in the column have relatively higher priority. For example, in the first column, if a property style (say background color) is set on an individual control type (say Buttons), then the value of this style property will have a higher priority than a value set for the same style property at the All controls level or the Style sheet level.

| Style sheet property of control set on... | Style sheet property of table set on... | Style sheet property of page set on... |
Style sheet: 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.

<table>
<thead>
<tr>
<th>Definition level in style sheet</th>
<th>When style sheet is set on page/table/control instance, style sheet applies to...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Page instance</td>
</tr>
<tr>
<td>Style sheet</td>
<td>Page instance; all tables and all controls on page</td>
</tr>
<tr>
<td>All controls</td>
<td>All controls on page</td>
</tr>
<tr>
<td>Control type</td>
<td>All controls of that type on page</td>
</tr>
<tr>
<td>Table</td>
<td>All tables on page</td>
</tr>
<tr>
<td>Page</td>
<td>Page instance</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition level in style sheet</th>
<th>When style sheet is set on page/table/control instance, style sheet applies to...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Table instance</td>
</tr>
<tr>
<td>Style sheet</td>
<td>Table instance; all controls in table</td>
</tr>
<tr>
<td>All controls</td>
<td>All controls in table</td>
</tr>
<tr>
<td>Control type</td>
<td>All controls of that type in table</td>
</tr>
<tr>
<td>Table</td>
<td>Table instance</td>
</tr>
<tr>
<td>Page</td>
<td>--</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition level in style sheet</th>
<th>When style sheet is set on page/table/control instance, style sheet applies to...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Control instance</td>
</tr>
<tr>
<td>Style sheet</td>
<td>Control instance</td>
</tr>
<tr>
<td>All controls</td>
<td>Control instance</td>
</tr>
<tr>
<td>Control type</td>
<td>Control instance if of that type</td>
</tr>
<tr>
<td>Table</td>
<td>--</td>
</tr>
<tr>
<td>Page</td>
<td>--</td>
</tr>
</tbody>
</table>

11.8.3 Priority across Style Sheets

The question of priority also arises when there are multiple definitions (at different levels, and in different style
sheets) for a single style property, and when more than one of these definitions applies to the style property of
a single design component. In this case, MobileTogether will look through definitions for this property at the
various style sheet levels in the order given below. The first one to match is the one that is used. The table
below uses the example of the *background color* property on a Button control.

<table>
<thead>
<tr>
<th>Definition level in style sheet</th>
<th>When style sheet is set on page/table/control instance, style sheet applies to...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Page instance</td>
</tr>
<tr>
<td>Style sheet</td>
<td>Page instance; all tables and all controls on page</td>
</tr>
<tr>
<td>All controls</td>
<td>All controls on page</td>
</tr>
<tr>
<td>Control type</td>
<td>All controls of that type on page</td>
</tr>
<tr>
<td>Table</td>
<td>All tables on page</td>
</tr>
<tr>
<td>Page</td>
<td>Page instance</td>
</tr>
</tbody>
</table>

*If the button control in the design references Stylesheet-1*

[Background color defined on the button control in the design](#) **Highest Priority**
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

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.
11.8.4 Applying User-Created Style Sheets

A user-created style sheet can be applied to page instances, table instances, and control instances. The style definitions in the user-created style sheet will be immediately applied to the selected design component and will override existing style definitions of a lower priority.

You can apply a user-created style sheet to a design component (page, table, or control) as follows:

1. In the design, select the design component (page, table, or control) to which you wish to apply a user-created style sheet.
2. In the Styles & Properties Pane, select the Style Sheet property of the page, table, or control to which you wish to apply the style sheet. In the screenshot below, the Style Sheet property of a control has been selected.

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: 

\[
\text{if } \left(\$MT_{\text{iOS}}=\text{true()}\right) \text{ then } '\text{iOSStyleSheet}' \text{ else } '\text{GeneralStyleSheet}'\]

Note: Switching style sheets often at runtime could cause solution execution to slow down.
11.8.5 Style Sheet Properties

In a style sheet, you can define styles for individual control types (such as Button controls and Label controls), tables, and pages. Select the design component for which you wish to define styles in the left-hand pane. The screenshot below shows that the Button control type has been selected. The properties of the selected design component appear in the right-hand pane. You can now select or enter values for individual properties. Click **Save** when done.

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.

Listing and grouping of properties

The context menu of a property (obtained by right-clicking the property) contains commands to list or group controls that are related to that property in the following ways:

- **List controls with the same direct style value**: Searches the design for the same property value as that of the currently selected property, and lists all controls that have this property value set at the control level. For example, if a label and a button have their Text Color property set to red, then both controls are listed when the command is executed for this property value.

- **List controls with the same style value**: Searches the design for the same property value as that of the currently selected property, and lists all controls that have this property value—whether the value is set at the control level or via a style sheet. For example, if a label and a button have their Text Color property set to red, and a style sheet containing a Text Color style of red is applied to a combo box, then all three controls will be listed when this command is executed for this property value.

- **Group controls by direct style value**: All controls in the design are grouped on the values of the currently selected property if these values are set on the control itself. For example, if the currently selected property is Text Color, then all controls that have the same Text Color value, set at the control level, are grouped together. So all controls that have, say, a Text Color value of red set at the control level are listed together, while those with another value are listed in their own group.

- **Group controls by style value**: All controls in the design are grouped on the values of the currently selected property—whether these values are set at the control level or via a style sheet. For example, if the currently selected property is Text Color, then all controls that have the same Text Color value, set at the control level or via a style sheet, are grouped together. So all controls that have, say, a Text Color value of red are listed together, while those with another value are listed in their own group.

The lists are displayed in the Listings Pane, and controls in these lists are hyperlinked.
11.9  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 markup.

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

11.9.1  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.

11.9.2 Rich Text Style Sheets: Setup

A Rich Text style sheet consists of multiple rules that describe the transformation, in both directions, between XML (data in the page source) and HTML (for the display in the Rich Text control). 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 to add an empty style sheet. Alternatively, you can add a style sheet with predefined styles for HTML elements by clicking the dropdown arrow of Add Style Sheet, and selecting Add HTML Style Sheet. See the next section below for a description of this style sheet.
3. Edit the name of the style sheet to something suitable for your project. In the screenshot below, for example, the style sheet has been renamed main.
4. Optionally, select a page source that has the element structure you want to use. (The elements of this page source will be used to provide the entry helper items that are displayed during editing.)
5. Click Save to save the style sheet with the project.
Predefined HTML style sheets
When adding a Rich Text style sheet, you can choose to add an HTML style sheet (see Point 2 of the procedure listed above). This style sheet contains predefined styles for the following elements (and also toolbar assignments):

- In the **Element Mappings** section, styles for the following commonly used HTML inline elements are defined: `b`, `strong`, `i`, `em`, `u`, `ins`, `del`, `mark`, `small`, `sub`, `sup`.
- In the **List Styles** section, styles for list-related HTML elements are defined.
- In the **Toolbar Assignments** section, the HTML elements `b`, `i`, `u`, `del` have been assigned to toolbar icons.

You can modify the existing styles and/or add new styles as required. How to do this is described in the section Rich Text Style Sheets: Styles.

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*. 
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:
  
  ```xml
  <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:
  
  ```xml
  <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.

![Context Menu Screenshot]

### 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.

![Editing Toolbar Screenshot]

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.

### 11.9.4 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.

![A Rich Text Example](image)

The control's toolbar

The toolbar of the Rich Text control (screenshot below) is displayed on Web and Windows clients only.

![Toolbar Example](image)
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 with a dropdown menu of markup tag sizes. You can select sizes from between a maximum, which shows the entire node name, to a minimum, which shows empty small start and end tags. The screenshot below shows markup tags at maximum size.

- Four combo boxes in which to select, respectively, the font family, font size, font color, and font background. The set of available fonts can be customized in the Browser Settings dialog. If no font is specified in the Browser Settings dialog, then the font selection combo box will be disabled in the deployed solution. A warning to this effect is displayed in simulations when the cursor is placed over the font selection combo box (see screenshot below).

- 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.
11.10 Solutions for Authenticated Users

Consider the following situation. An end user is authenticated (correctly logged in) on one MobileTogether Server and is running a solution from that server. You, as the designer, wants to start a solution from another MobileTogether Server on that user's device. Since the user is already authenticated on one MobileTogether Server, you would like the solution on the second MobileTogether Server to be started directly, without the user having to log in to the second MobileTogether Server. MobileTogether enables you to securely pass authentication information to a solution on another MobileTogether Server.

Note: The authentication described in this topic applies only to solutions running on web clients.

Setting up authenticated users for remote solutions

The steps for setting up the transfer of user authentication to a solution on another MobileTogether Server are given below.

Terminology note

- Authentication Host: The first MobileTogether Server, on which the authentication has already been carried out.
- Authentication Receiver: The second MobileTogether Server, from which the second solution is served and on which we want authentication for the solution to be automatic.

1. In the MobileTogether Server settings of the Authentication Receiver, enter and enable the settings of the Authentication tab. These settings are: (i) the address of the Authentication Host, (ii) the secure HTTPS port for mobile clients, and (iii) the Audience string, which is a unique string that identifies the audience of this MobileTogether Server. Note also: (i) that both servers must use SSL encryption (HTTPS connections), and (ii) that both solutions (the calling and called) must be run for anonymous users.

2. In the calling solution, define a Solution Execution action (see screenshot below) at an appropriate point during solution execution.

The relevant settings of the action are:
• **The solution's address:** Which should evaluate to an address of the form
  \url{https://MTServerAddress/run?d=/public/SolutionName}.

• **Token:** A user-defined XML tree that is securely passed to the solution on the second server. You
define this XML tree so that it contains all the information you want to pass. MobileTogether requires
only that this entry is a well-formed XML tree.

• **Audience:** This string must match the *Audience* setting of the Authentication Receiver (*see Point 1
above*).

3. If the audience matches, then the XML tree that is passed to the Authentication Receiver (via the
*Token* setting of the Execution Solution action; *see previous point*) will be passed to the second
solution in the `$MT_AuthenticationToken` variable. You can access nodes in the token via XPath
expressions that use the variable (for example: `$MT_AuthenticationToken/Root/User/@id`). If the
audience does not match, then the solution will start, but without a valid authentication token. It is left
to you as the developer to decide how to handle this situation. One way would be to show a suitable
message and forward the user back to the calling solution.
11.11 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=3YAaVdDDDA4SYysGOMm4A4

**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.

```xml
<mt-run-solution-url
    ServerAddress? as xs:string,
    SolutionName? as xs:string,
    InputParameters? as xs:string
    as xs:string

mt-run-solution-url

Generates a URL to open the specified solution in a MobileTogether client. When the URL is tapped, the Altova MobileTogether Client app is opened and the solution is started in the app. The URL is generated from either (i) the function's three submitted arguments (listed below), or (ii) the function's InputParameters argument.
```
- **ServerAddress**: Takes the name or IP address of the MobileTogether Server on which the solution that you want to run is deployed. If this argument is omitted or it is the empty string, then the current server is used.
- **SolutionName**: Takes the deployed path of the solution on the server. For example: `/public/MySolution` (which would point to the `MySolution.mtd` file in the `/Public` container). If this argument is omitted or it is the empty string, then the current solution is used.
- **InputParameters**: Takes the function `mt-run-solution-url-parameters` as its input. The argument of the `mt-run-solution-url-parameters` function is either (i) a sequence of string values that will be the values of the query’s parameters, or (ii) a map of *key*:value pairs that provide the name and value of the respective parameters. This function returns a string that is correctly encoded and percent-escaped according to the rules for encoding URL query strings. See the description of the `mt-run-solution-url-parameters` function below. (Additionally, the `InputParameters` argument can be provided as a string that is already encoded for the query string part of a URL (see fourth example below).)

The `mt-run-solution-url` function therefore creates a URL, with or without query parameters, that accesses a solution on a MobileTogether Server. The query parameters are passed to the solution when the solution is opened via the URL. The values of these parameters can be accessed in other design components by using the `$MT_InputParameters` global variable.

### Examples

- `mt-run-solution-url('100.00.000.1', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the server with the IP address `100.00.000.1`. The URL has no query parameters.
- `mt-run-solution-url('', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the current server. The URL has no query parameters.
- `mt-run-solution-url('', '', mt-run-solution-url-parameters(('2015', 'USA', 'true')))` returns a URL that points to the current solution on the current server. The argument of the `mt-run-solution-url-parameters` function in this example is a sequence of string values that will be the values of the query’s parameters. The first string will be the value of the first parameter, the second string will be the value of the second parameter, and so on. See the description of the `mt-run-solution-url-parameters` function below.
- `mt-run-solution-url('', '', 'in1=value1&in2=value2%3FAndMoreValue2')` returns a URL that points to the current solution on the current server. The `InputParameters` argument is submitted as a string already encoded as a URL query string.

Note the following points:

- The first argument, **ServerAddress**, is used to look up information on the client about a server having the submitted name/address. The port number, user name, and user password that are associated with the server name are then used to connect to the server. So if a URL is generated with a server name that is not recognized by the client, then the URL will not work.
- The second argument, **SolutionName**: (i) generates the deployed path (on the server) if the solution is run on the server, but (ii) generates a file path for simulations.
- The third argument, **InputParameters**, uses the MobileTogether-specific XPath extension function called `mt-run-solution-url-parameters` to generate and encode the query’s parameter-value pairs. Do not confuse the `mt-run-solution-url-parameters` function (which encodes the query parameters) with the `mt-run-solution-url` function (which generates the whole URL).
Using hyperlink query parameter values in other design components

When a solution is opened by triggering a hyperlink that is associated with a control event or page event, any parameter values in the hyperlink's URL are passed to the solution and can then be used in other design components in the target solution. The values are stored by default as a map in the $MT_InputParameters global variable of the target solution.

Alternatively, you can change the data structure of the $MT_InputParameters variable in individual projects (in the More Project Settings dialog) to be a sequence of string values. If string values are passed to $MT_InputParameters, these are alphabetically sorted on the key of the URL's parameters. The order of the string values in the $MT_InputParameters sequence is the same as that in the sequence submitted to the mt-run-solution-url-parameters function for generating the URL's query parameters. Since the order of string values in the $MT_InputParameters is known to you (alphabetically sorted on the parameter keys), each string can be accessed in XPath expressions by using position predicates. For example: $MT_InputParameters[1] returns the first string value in the sequence, and $MT_InputParameters[2] returns the second string value.

Linking to a solution from an email that the end-user sends

The Send Email To 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).

![Screenshot](image)

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 (XX) 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.
11.12 Sizes: Pixels, DPI, DP, SP

The size of objects and text in the design can be specified in pixels (px). However, the display on the client device depends not only on the specified pixel size but also on the device resolution, and, in the case of text, additionally on the text size selected by the device user. See the first row of the diagram below.

The new units of device-independent pixels (dp) and scale-independent pixels (sp), however, enable you to obtain a display of the same size across devices that have different resolutions. The diagram below shows how resolution affects pixel sizes and dp sizes.
**Red square in different resolutions: 2px wide, 2px high**

- Resolution = 1 dpi
- Resolution = 2 dpi
- Resolution = 4 dpi

**Red square in different resolutions: 1dp wide, 1dp high**

- 1dp = 1px
- 1dp = 2px
- 1dp = 4px

\[ px = dp \times \left(\frac{dpi}{160}\right) \]
**Pixels and resolution**
The resolution of a screen is the number of pixels in an inch of screen length; its unit is **ppi** (pixels per inch), more commonly referred to as **dpi** (dots per inch). So if the pixel density of a screen (its dpi) is high, then an object with the same pixel size will be displayed smaller on higher resolution screens. This can lead to the same object being displayed in different sizes on screens that have the same dimensions but different resolutions. In the top row of the diagram above, for example, all the rectangles (representing devices) have the same width and height, but different resolutions. As a result, the red square (width=2px and height=2px) is progressively smaller in the higher resolution rectangular screens. For example, the rightmost screen has a resolution of 4 pixels per inch; as a result, the red area of 2x2 pixels covers a smaller area of this screen. This difference in area size can be overcome by using device-independent pixels (**dp**) as the unit of length.

**Device-independent pixels (dp)**
If device-independent pixel (**dp**) is used as the unit of length, then the operating system of the device maps the dp value to a corresponding number of pixels based on the resolution of the device screen. For this mapping, 1 dp is considered to be equal to 1 pixel on a 160 dpi resolution screen. The corresponding number of pixels can be calculated with the formula $px = dp \times (dpi/160)$. By using device-independent pixels, you will have better control over your design across devices of differing resolutions.

**Scale-independent pixels (sp)**
A scale-independent pixel (**sp**) is the same as a device-independent pixel (**dp**), with an additional scaling factor that is based on the font size that the user selects in the device's system settings. Scale-independent pixels should be used only as a unit for text. Avoid using **sp** as a unit for non-text components.

**Note:** DP and SP sizes in the designer's device simulations will not exactly match sizes on the actual client devices.

**XPath extension function: mt-convert-units**
MobileTogether has a built-in XPath extension function, which you can use for converting between the three values:

```
mt-convert-units
```

Converts the length value specified in the **Size** argument to an equivalent value in the unit specified by the **TargetUnit** argument. You can convert between any two of the following units: **px**, **dp**, and **sp**. Both input arguments, as well as the output value, are strings. For more information about units and conversion between them, see [Sizes: Pixels, DPI, DP, SP](#).

**Examples**
- `mt-control-unit("24px", "dp")` returns, depending on the device's resolution, for example, "22dp" on one device and "20dp" on another
- `mt-control-unit("20sp", "px")` returns, depending on the device's resolution, for example, "22px" on one device and "24px" on another
12 XPath/XQuery Expressions and Functions

XPath/XQuery expressions play an important role in MobileTogether designs, and MobileTogether Designer provides (i) powerful features to help you edit and debug expressions, as well as (ii) a range of extension functions. This section describes the following:

- **XPath/XQuery Window**, in which you can build expressions easily, run test evaluations of the expressions, and debug the expressions using a range of diagnostic features
- **MobileTogether Extension Functions**
- **User-Defined XPath/XQuery Functions**
- **FAQ about XPath/XQuery**
12.1 XPath/XQuery Window

The **XPath/XQuery Window** *(screenshot below)* is accessed whenever an XPath expression needs to be entered or edited, and it is used to correctly create and to test XPath expressions for a range of MobileTogether features.

- **Expression Builder** provides entry helpers, popup descriptions of context-sensitive XPath/XQuery constructs, and auto-completion. To select Expression Builder, click **Builder** in the toolbar *(see screenshot above)*.
- **Expression Evaluator** gives you a preview of expression results, thus enabling you to review and correct your expression. To select Expression Evaluator: Click **Evaluator** in the toolbar, and then select **Start Evaluation** in the dropdown menu of **Start Evaluation/Debugging** *(see screenshot above)*. Alternatively, simply select **Start Evaluation**.
- In **XPath Debugger**, you can step through the evaluation of an XPath expression to see how the expression is processed. You can subsequently modify the expression as needed. To select XPath Debugger: Click **Evaluator** in the toolbar, and then elect **Start Debugging** in the dropdown menu of **Start Evaluation/Debugging** *(see screenshot above)*. Alternatively, simply select **Start Debugging**.
Availability of the XPath/XQuery Window

The XPath /XQuery Window is available in the following contexts:

- In the design, it is available wherever an XPath expression may be entered, for example, when entering expressions to set the values of styles and properties.
- From the simulator. In this context, it is useful for analyzing how the solution will behave in various runtime situations and with different datasets.

12.1.1 Expression Builder

The Expression Builder enables you to build correct XPath expressions with respect to the current context node. It enables you to enter correct node locator paths, select nodes in the schema tree, and enter operators and functions that are syntactically correct. Switch to Expression Builder by clicking the Builder button (see screenshot below). When the Builder button is clicked, entry helper panes to help you build an XPath expression become visible. Double-click an entry in the entry helpers to enter it at the current cursor location in the Expression pane. On clicking OK, the expression is entered at the current location in the design.
There are three entry helper panes:

- A schema tree for selecting and entering element and attribute nodes. If the Relative XPath check box is checked (as in the screenshot above), then the path to the selected node is entered relative to the context node—which is the node in the design within which the XPath expression is being built. The context node is shown below the schema tree pane. You can see in the screenshot above that the context node is the Customer element. You can also see, from the title bar of the window, that this XPath expression is being built for the Text property of the Label3 control. If the Relative XPath check box is unchecked, then an absolute XPath expression to locate the selected node is entered; such an expression would start at the document root, for example: $CUSTOMERS/CUSTOMERS/Customer/Name.

- An entry helper pane for operators and expressions. These include: (i) axes (ancestor::, parent::, etc), (ii) operators (for example eq and div), and (iii) expressions (for # in # return #, etc). The items of the pane can be either listed alphabetically or grouped by functional category. Select Hierarchical (for the grouped option) or Flat (for an alphabetical listing) from the dropdown menu in the title bar of the pane.

- An entry helper with XPath functions either listed alphabetically or grouped by functional category. Select Hierarchical (for the grouped option) or Flat (for an alphabetical listing) from the dropdown menu in the title bar of the pane. The Names/Types option enables you to choose whether the function's
arguments are displayed as names or datatypes. Note that the list of available functions includes MobileTogether extension functions and Altova extension functions.

Features of the Builder

- To view a text description of an item in the Operators pane and Functions pane, hover over the item.
- Each function is listed with its signature (that is, with its arguments, the datatypes of the arguments, and the datatype of the function's output).
- The signatures of functions are given with either the names or the datatypes of the function's arguments and output. Select Names or Types from the dropdown menu in the title bar of the pane for the respective option.
- Double-click an item in any of the three entry helper panes (operator, expression, or function) to insert that item at the cursor location in the expression. Functions are inserted with their arguments indicated by placeholders (the # symbol).
- If (i) text is selected in the XPath expression's edit field (for example Name in the screenshot above), and (ii) an expression or function that contains a placeholder is double-clicked to insert it, then the text that was selected is inserted instead of the first placeholder.
- After you have entered a function in the expression, hovering over the function name displays a popup containing the function's signature and a text description of the function. If additional signatures exist for a function, then this is indicated by an overload factor at the bottom of the popup. (Note: If multiple functions have the same name but each takes a different set of arguments, then, this set of functions is considered, more correctly, to be a single function with multiple signatures.)
- If, in the Expression pane, you place the cursor within the parentheses of a function and press Ctrl+Shift+Spacebar, you can view the different signatures of that function.

Building XPath expressions

The Edit XPath Expression dialog helps you to build XPath expressions in the following ways.

- **Title bar of window displays**
  The title bar of the XPath/XQuery window displays information about the context (such as the design component (project, page, control, etc) and property) for which the expression is being built. For example, in the screenshot above, the expression is being built for the Text property of the Label3 control on a page named Customers.

- **Context node and schema tree**
  The Context text box immediately below the Schema Tree pane shows you the context node of the expression. In the schema tree itself, you can see where the context node occurs and quickly build the XPath expression by referring to the schema tree.

- **Inserting a node from the schema tree**
  Double-click a node in the schema tree to insert it in the XPath expression. If the Relative XPath check box is checked, then the node is inserted with a location path that is relative to the context node. For example, in the screenshot above, the Name element, which is a child of the Customer element (the context node), has been inserted with a path that is relative to the Customer element. If the Relative XPath check box is not checked, then the Name node would be inserted as $CUSTOMERS/Customers/Customer/Name.

- **Inserting XPath axes, operators and expressions**
  The Select Operator/Expression pane lists the XPath axes (ancestor::, parent::, etc), operators (for example, eq and div), and expressions (for # in # return #, etc). The display can be toggled between an alphabetical listing and a hierarchical listing (which groups the items according to
functionality). To insert an axis or operator in the XPath expression, double-click the required item. Placing the mouse cursor over an axis/operator/expression displays a brief description of that item.

- **Inserting XPath functions**
  The Select Function pane lists XPath functions alphabetically or grouped according to functionality (select Hierarchical or Flat at the top of the pane to switch between the two arrangements). Each function is listed with its signature. If a function has more than one signature, that function is listed as many times as the number of signatures. Arguments in a signature are separated by commas, and each argument can have an occurrence indicator. The symbol ? indicates a sequence of zero or one items of the specified type; * indicates a sequence of zero or more items of the specified type; + indicates a sequence of one or more items of the specified type. The arguments can be displayed as names or as datatypes. Select the appropriate option (Names or Types) at the top of the pane to toggle between the two display options. Each function also specifies the return type of that function. For example: => date ? indicates that the expected return datatype is a sequence of zero or one date items. Placing the mouse cursor over a function displays a brief description of the function.

**Intelligent editing during direct text entry**
If you type an expression directly in the Expression text box, a list of options that are available at that point are displayed in a popup (see screenshot below).

These include the following components:

- elements (such as presswatch in the screenshot above)
- descendant nodes (presswatch/selection in the screenshot above), and parent node of the context node
- XPath functions (fn:upper-case above) and XPath axes (ancestor-or-self above)
- a list of the global variables 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.

**12.1.2 Expression Evaluator**

The Expression Evaluator (screenshot below) enables you to apply an XPath/Query expression to a data source file and preview the return value. In Expression Evaluator, the window has three panes:

- an Expression pane (at top left), in which you enter the XPath/XQuery expression to evaluate
- a Schema Tree pane (top right), which shows the structure of the document that is currently loaded; the pane also acts as an entry helper for adding nodes from schema trees
- a Results pane (bottom), which displays the result of the evaluation
The XPath expression in the screenshot above is applied to the tutorial file `SubpagesAndVisibility.mtd`.
You can open this design and try out the XPath expression (listed below) during a simulation (started with F5).

### Listing of XPath expression shown in screenshot above

```xml
for $i in $CUSTOMERS/Customer,
  $j in $ORDERS/Order[CustomerCode=$i/@code]
return
concat($i/Name,
  ', ID=', $i/@code,
  ', Order=', $j/@number,
  ', Amount=', $j/OrderAmount),
for $i in $CUSTOMERS/Customer
return
concat($i/Name, ', TotalAmount=',
  sum($ORDERS/Order[CustomerCode=$i/@code]/OrderAmount))
```
Using Expression Builder and switching to Expression Evaluator for the results

- You can switch between the Builder and the Evaluator without losing the expression that was typed in the Expression pane. To switch, click the appropriate toolbar button (Builder or Evaluator).
- The advantages of the Builder are entry helper support and the simultaneous availability of schema trees of all page sources. (In the Evaluator, on the other hand, only one schema tree is available at a time.)
- After building an expression, you can evaluate it in Expression Evaluator by selecting Start Evaluation.
- If the Evaluate on Typing button in the window's toolbar is toggled on, then the expression is evaluated as you type in Expression Evaluator.

Data source on which the expression is applied

Two situations may be distinguished in this context:

- If the XPath/XQuery Window is opened from the Simulator window, then the data sources of the design have already been loaded for the simulation. As a result, evaluation can be directly carried out on these data sources. You can see the structures (schema trees) of these data sources in the Schema Tree pane of both the Evaluator and Builder. (In Expression Evaluator, however, only one schema tree is displayed at a time (see screenshot above). To switch trees in the Schema Tree pane of Expression Evaluator, select another tree in the dropdown menu of the schema tree selector (see screenshot above).
- If the XPath/XQuery Window is opened from within the design and no corresponding XML file is loaded, this is indicated (see screenshot below). In this case, you must explicitly load an XML file that has the same structure as one of the page sources of the design. Do this via the Load button below the Schema Tree pane (see screenshot above that shows Schema Tree).

Schema Tree pane

In addition to displaying the selected schema tree (described above), the Schema Tree pane provides the following features:

- The icons below the pane toggle on/off the display of the following XML syntactic constructs: (i) processing instructions, (ii) comments, (iii) attributes, (iv) text nodes. You can therefore see the entire XML document structure, together with the text contents of nodes, but you can also hide certain constructs if you wish to reduce clutter in the pane.
- You can change the context node of the XPath expression by clicking the node in the document tree that you want as the new context node and then selecting Set Evaluation Context. (located below the Schema Tree pane at right). Note, however, that the actual context node for the expression at run time will be the context node within which the current design component is being created.
Expression pane
The XPath/XQuery expression is entered in the Expression pane. The results of the evaluation are displayed in the Results pane (see screenshot above).

Note the following points:

- To create the expression over multiple lines (for easier readability), use the Return key.
- To increase/decrease the size of text in the expression field, click in the expression field, then press Ctrl and turn the scroll wheel. Note that this also applies in the Results pane.

Results pane
The Results pane is shown in the screenshot below, at bottom. Note that it has its own toolbar.

The Results pane has the following functionality:
The result list consists of two columns: (i) a node name or a datatype; (ii) the content of the node.

If the XPath expression returns nodes (such as elements or attributes), you can select whether the entire contents of the nodes should be shown as the value of the node. To do this, switch on the toggle *Show Complete Result*.

For data sources loaded during simulations: When the result contains a node (including a text node)—as opposed to expression-generated literals—clicking that node in the Results pane highlights the corresponding node in the Schema Tree pane.

You can copy both columns of a result sub-line, or only the value column. To copy all columns, right-click a sub-line and toggle on *Copying Includes All Columns*. (Alternatively you can toggle the command on/off via its icon in the toolbar of the Results pane.) Then right-click the sub-line you want to copy and select either *Copy Subline* (for that subline) or *Copy All* (for all sublines).

**Toolbar of the Results pane**
The toolbar of the Results pane contains icons that provide navigation, search, and copy functionality. These icons, starting from the left, are described in the table below. The corresponding commands are also available in the context menu of result list items.

<table>
<thead>
<tr>
<th>Icon</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next, Previous</td>
<td>Selects, respectively, the next and previous item in the result list</td>
</tr>
<tr>
<td>Copy the selected text line to the clipboard</td>
<td>Copies the value column of the selected result item to the clipboard. To copy all columns, toggle on the <em>Copying includes all columns</em> command (see below)</td>
</tr>
<tr>
<td>Copy all messages to the clipboard</td>
<td>Copies the value column of all result items to the clipboard, including empty values. Each item is copied as a separate line</td>
</tr>
<tr>
<td>Copying includes all columns</td>
<td>Switches between copying (i) all columns, or (ii) only the value column. The column separator is a single space</td>
</tr>
<tr>
<td>Find</td>
<td>Opens a <em>Find</em> dialog to search for any string, including special characters, in the result list</td>
</tr>
<tr>
<td>Find previous</td>
<td>Finds the previous occurrence of the term that was last entered in the <em>Find</em> dialog</td>
</tr>
<tr>
<td>Find next</td>
<td>Finds the next occurrence of the term that was last entered in the <em>Find</em> dialog</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the result list</td>
</tr>
<tr>
<td>Collapse multi-line results to a single line</td>
<td>If the value column of a result item contains multi-line text (text that includes newline character/s), you can toggle between a multi-line and single-line display</td>
</tr>
<tr>
<td>Show complete result</td>
<td>Shows the entire content of nodes as the value of the node</td>
</tr>
</tbody>
</table>

**12.1.3 XPath Debugger**

The XPath Debugger of the XPath/XQuery Window enables you to debug XPath expressions in the context of the loaded file/s. You can open the XPath/XQuery Window via the following entry points:

- In the design, it is opened wherever an XPath expression may be entered or edited: for example, when entering expressions to set the values of *styles and properties*.
- From the simulator, it is opened by clicking the *Evaluate XPath* button in the Page Sources pane.
When the simulator stops at an action, you can start XPath Debugger by clicking the Step Into XPath button of the Actions Debugger.

**Note:** In the first two contexts listed above, you can enter any XPath expression you like and debug it against any XML file, which you must load into the Debugger. In the third context, however, you will be debugging the XPath expression that is specific to the current action, and you will be debugging this expression against the current action's page sources; consequently, the option to load an XML file is disabled.

The descriptions and screenshots in this section refer to the XPath Debugger when accessed via the Simulator's Page Sources pane. The Debugger behaves in the same way irrespective of the context from which it was launched. Note, however, that the option to load an XML file is disabled when the Debugger is started from the third context.

**Starting XPath Debugger**

To start XPath Debugger, select **Start Debugging** in the **Start Evaluation/Debugging** dropdown menu (screenshot below). You can switch between the Builder (for help with building the expression) and the Debugger.

Either before or after entering an XPath/XQuery expression, select an XML file on which the expression is to run (see **Running the Debugger** below). Start debugging by clicking **Start Evaluation/Debugging (F5)**. The Debugger will run the expression on the loaded XML file and display results in the panes in the lower part of the window.

**Buttons for setting up XPath Debugger**

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Start" /></td>
<td><strong>Start Evaluation/Debugging (F5)</strong> Starts the debugger</td>
</tr>
<tr>
<td><img src="image" alt="Switch" /></td>
<td><strong>Switch to Builder</strong> Switches to Expression Builder mode, which provides context-sensitive entry helpers to help construct expressions</td>
</tr>
<tr>
<td><img src="image" alt="Evaluation" /></td>
<td><strong>Evaluation on typing</strong> Switches on the evaluation of expressions while the expression is being typed</td>
</tr>
</tbody>
</table>

**Layout of XPath Debugger**

XPath Debugger has two panes that are additional to the Results pane of the Evaluator (see screenshot below):

- the Watch Expressions and Variables pane; both, variables and watch expressions, are shown in the same pane
- the Call Stack and Debug Points pane, each of which has a separate tab on the right-hand side of the pane
XPath Debugger provides the following features:

- Enables you to step into the XPath evaluation process, one step at a time to see how the XPath expression is being evaluated. Use the Step Into (F11) toolbar button for this. At each evaluation step, the part of the expression being currently evaluated is highlighted in yellow (see screenshot above), while the result of evaluating that step is shown in the Results pane. For example, in the screenshot above, all the Order elements that have a descendant element named CustomerCode containing the value ‘456’ have been selected.

- Set breakpoints where you want to pause the evaluation and check results. Use the Start Debugging (F5) toolbar button to go through the evaluation, pausing only at breakpoints. Using breakpoints is quicker than pausing at every processing step with Step Into (F11).
• Set tracepoints to see the evaluation results at the steps marked as tracepoints. The evaluation will not pause—they pause only at breakpoints—but the tracepoint results will be displayed in a list in the Results pane.
• Watch expressions are XPath expressions that are evaluated at every step of debugging. You can use them to check information you want at each step, such as document data or aspects of the evaluation. Watch expressions are very useful when used together with breakpoints.
• Variables that are in scope at the current step, including their values, are displayed in the Watch Expressions and Variables pane.
• Processor calls of an evaluation step are shown in the Call Stack tab of the Call Stack and Debug Points pane.
• If breakpoints and tracepoints have been set, then these are displayed in the Debug Points tab of the Call Stack and Debug Points pane.

For more information about these features, see their descriptions below.

Running the Debugger
The broad steps for debugging an XPath expression are as follows:

1. Enter the XPath expression in the expression pane.
2. Make sure that the XML file on which you want to apply the XPath expression is loaded. If it is not loaded, then use the Load button (located below the Schema Tree pane; see screenshot above) to browse for the file and load it.
3. Set any breakpoints or tracepoints you want. A breakpoint is a point at which the evaluation is paused. A tracepoint is a point in the evaluation at which the result is recorded; tracepoints thus provide a traceable path of evaluation results.
4. If you click Start Debugger, evaluation is paused at the first breakpoint. Click Start Debugger repeatedly to progress through each breakpoint to the end of the evaluation.
5. Use the Step Into/Out/Over functionality to go step-by-step (as opposed to breakpoint-by-breakpoint) through the evaluation.
6. You can also run the debugger up to a point in the XPath expression: (i) Mark the point by placing your cursor at that location, and (ii) click the Run to Cursor toolbar icon.

Buttons for debugging

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Start Debugger (F5)" /></td>
<td>Starts the debugger. Evaluation pauses at the next breakpoint</td>
</tr>
<tr>
<td><img src="image" alt="Stop Debugger (Shift+F5)" /></td>
<td>Exits the evaluation and stops the debugger</td>
</tr>
<tr>
<td><img src="image" alt="Step Into (F11)" /></td>
<td>Proceeds through the evaluation, one step at a time.</td>
</tr>
<tr>
<td><img src="image" alt="Step Out (Shift+F11)" /></td>
<td>Steps out of the current evaluation step, and goes to the parent step</td>
</tr>
<tr>
<td><img src="image" alt="Step Over (Ctrl+F11)" /></td>
<td>Steps over descendant steps</td>
</tr>
<tr>
<td><img src="image" alt="Run to Cursor (Ctrl+F5)" /></td>
<td>Starts the debugger and runs it to the cursor location in the XPath expression</td>
</tr>
<tr>
<td><img src="image" alt="Insert/Remove Breakpoint (F9)" /></td>
<td>Inserts/removes a breakpoint at the expression step where you place the cursor</td>
</tr>
</tbody>
</table>
Stepping into, out of, and over evaluation steps
The Step Into functionality enables you to go step-by-step through the evaluation. Each click of this command takes you through the next step of the evaluation; the current step is shown by the highlighting in the expression (see screenshot below). The Step Out functionality takes you to a step on a higher level as the current step, whereas the Step Over functionality steps over lower-level steps and takes you to the next step on the same level. You can try out the Stepping functionality by using the expression shown in the screenshot below ($ORDERS/Orders/Order[CustomerCode="456"]/@number) and clicking the three Step buttons to see how they work. (The XML file in this example is a page source of the tutorial SubPages-And-Visibility. Open this file and do the following: (i) Start the Simulator; (ii) On the top page that appears in the Simulator, click Show All Orders, (iii) In the Page Sources pane of the Simulator, click the Evaluate XPath toolbar icon, (iv) switch to the Debugger (see above).)

The screenshot below shows the evaluation when processing has been paused on reaching the locator step Order[CustomerCode='456']. At this step, the result shows the two Order elements that each have a CustomerCode child element containing the value '456'. The two elements (with their entire node contents) are displayed in the Results pane.
Breakpoints

Breakpoints are points where you want the Debugger to stop after it has been started with Start Debugger. They are useful if you wish to analyze a specific part of the expression. When the Debugger stops at the breakpoint, you can check the result and could then use the Step Into functionality to display the results of the next steps of the evaluation. To set a breakpoint, place the cursor in the expression at the point where you want the breakpoint, and click the Insert/Remove Breakpoint (F9) toolbar button. The breakpoint will be marked with a dashed red overline. To remove a breakpoint, select it and click Insert/Remove Breakpoint (F9).

Note: You can make a breakpoint conditional by entering the condition on the breakpoint's listing in the Debug Points pane. See Debug Points below.
Note: You can start XPath Debugger directly from a simulation to debug XPath expressions that have breakpoints set on them. To do this start MT Debugger in Breakpoints Mode.

Tracepoints
Tracepoints are points at which the evaluation results are recorded. These results are displayed in the Traces tree of the Results pane (see screenshot below). Tracepoints enable you to see the evaluation results of particular parts of the expression. For example, in the screenshot below, tracepoints have been set on the OrderAmount node. The results at these tracepoints are shown in the Traces tree.

To set a tracepoint, place the cursor at the point where you want the tracepoint, and click the toolbar button Insert/Remove Tracepoint (Shift+F9). The tracepoint will be marked with a dashed blue overline (see screenshot below). To remove a tracepoint, select it and click Insert/Remove Tracepoint (F9).

Note: If both a breakpoint and a tracepoint are set on the same part of an expression, then the overline is composed of alternating red and blue dashes.

Also see Debug Points below.

Watch Expressions, Variables and Call Stack
Watch expressions and variables are displayed in the Watch Expressions and Variables pane (bottom center pane in the screenshot below). In this pane, you can filter the view to show/hide (i) the current context item, (ii)
local variables, (iii) global variables. Do this by toggling on/off the respective toolbar icons. The current context item is the node that is currently being evaluated.

**Watch expressions**

Watch expressions are XPath expressions that you can enter, either before evaluation starts or during a pause in evaluation. The expression is evaluated in the context of the current node at each evaluation step where the debugger stops. Watch expressions can be used for the following purposes:

- To test specific conditions. For example in the screenshot above, the watch expression `$i/CustomerCode="789"` is used to test whether the current order element has a `CustomerCode` with a value of "789". The result `false` in the case of the first order element tells us that this order does not have a `CustomerCode` value of "789". (The value is "456" as can be seen in the Results pane.)
- To find data within a certain context. For example, in the screenshot above, we have entered the watch expression `$i/CustomerCode, $i/OrderDate` to look up these details of the current order.
- To generate additional data. For example, in the screenshot above, we have entered the watch expression `count($ORDERS//Order)` to count all the order elements.

To enter a watch expression, click **Add Watch** in the pane's toolbar, then double-click the new watch entry to enter the XPath expression and click **Enter** when done. To remove a watch expression, select it and click **Remove Watch** in the toolbar. If, during debugging, the expression cannot be correctly evaluated for some reason (for example, if one of its variables is out of scope), then the watch expression turns red.
Variables
Variables that have been declared in the expression and that are in scope in the current evaluation step will be displayed together with their respective current values. They are displayed below the watch expressions.

In the screenshot above, for example, processing has been paused at the breakpoint on the variable $i. The $i variable is in scope at this evaluation step. So $i is displayed with its current value, which in the screenshot above is the first Order element. Since the document's root node, $ORDERS, is also a variable, it is also displayed with its content, which is the document root.

Call stack
The Call Stack tab of the Call Stack and Debug Points pane (bottom right pane in the screenshot above) displays the processor calls up to that point in the debugging. The current processor call is highlighted in yellow. Note that only the calls that directly led to the current evaluation step are displayed.

Debug Points
The Debug Points tab of the Call Stack and Debug Points pane (bottom right pane in the screenshot below) shows the breakpoints (with solid red circles) and tracepoints (solid blue circles) that you have set on the expression. Each debug point is listed with its line and character number. For example, AxisStep@2:12 means that there is a debug point on line 2, character 12 of the expression in the expression pane.

To stop at each breakpoint in the expression, click Start Debugger (F5).

Note the following features:
You can make each breakpoint conditional on some factor by entering a **break condition** for it. Do this in the Debug Points pane by (i) double-clicking the breakpoint's *Enter break condition* entry, (ii) entering the expression for the condition, and (iii) pressing *Enter*. That breakpoint will be enabled only when the condition evaluates to **true**. For example, in the screenshot above, the break condition `$i/CustomerCode="456"` will enable the breakpoint on `$i` (that is, the current order element) when the Order element's CustomerCode child element has a value of "456". In effect, processing will pause only at those orders that have a customer code of **456**. The screenshot shows the evaluation paused at the sixth order, which is such an order. The breakpoint is not triggered for orders with other customer codes. (You can set a watch expression, as in the screenshot above, to show which orders have a customer code of **456**: the orders with the numbers 001 and 006.)

- You can enable/disable all debug points by clicking their respective toolbar buttons: **Enable All Debug Points** and **Disable All Debug Points** *(buttons encircled in green in the screenshot above)*. When a debug point is disabled, it is deactivated for all evaluations till it is enabled again.
- You can enable/disable individual breakpoints in their respective context menus.
- Tracepoints are simply listed with their details. They can be enabled/disabled.
- If you click **Clear All Debug Points**, then all breakpoints and tracepoints in the design will be removed.
- If an error occurs, for which no handling has been defined, this can be picked up. Simply switch on the **Break on Unhandled Error** toggle command in the toolbar. The Debugger will break on such errors.

### Toolbar commands in panes

The panes of the XPath/XQuery Window in Debug Mode *(see screenshot above)* contain buttons that provide navigation, search, and copy functionality. These buttons, starting from the leftmost in each pane, are described in the table below. The corresponding commands are also available in the context menu of listed items.

<table>
<thead>
<tr>
<th>Icon</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next, Previous</td>
<td>Selects, respectively, the next and previous item in the result list</td>
</tr>
<tr>
<td>Copy the selected text line to the clipboard</td>
<td>Copies the value column of the selected result item to the clipboard. To copy all columns, toggle on the <em>Copying includes all columns</em> command <em>(see below)</em></td>
</tr>
<tr>
<td>Copy all messages to the clipboard</td>
<td>Copies the value column of all result items to the clipboard, including empty values. Each item is copied as a separate line</td>
</tr>
<tr>
<td>Copying includes all columns</td>
<td>Switches between copying (i) all columns, or (ii) only the value column. The column separator is a single space</td>
</tr>
<tr>
<td>Find</td>
<td>Opens a <em>Find</em> dialog to search for any string, including special characters, in the result list</td>
</tr>
<tr>
<td>Find previous</td>
<td>Finds the previous occurrence of the term that was last entered in the <em>Find</em> dialog</td>
</tr>
<tr>
<td>Find next</td>
<td>Finds the next occurrence of the term that was last entered in the <em>Find</em> dialog</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the result list</td>
</tr>
</tbody>
</table>

### Closing the XPath Debugger

To close the XPath Debugger, click **Stop Evaluation/Debugging**.
12.2 MobileTogether Extension Functions

The following XPath extension functions, created specifically for use in MobileTogether designs, can be used in XPath expressions anywhere in the design. The XPath default namespace is used for calls to these extension functions.

Updating the return values of XPath functions

An XPath function is evaluated only when its containing XPath expression is evaluated. This typically happens when an action that contains the XPath expression is triggered or when a data change causes an XPath expression to be evaluated.

For example, consider an XPath expression that contains the function `mt-audio-is-playing`. This function returns either `true` or `false`. When the expression is evaluated at some given time, let us say that the return value is `true` (because audio is playing). If this value is displayed in the solution, then the value will not automatically change when the audio stops playing. For this to happen, the function will have to be called again so that the new value updates the value in the display.

The number of ways in which such values can be updated depends on the particular design mechanisms that have been used. One possible way to update such values is via the timer of the `OnPageRefresh` 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.)

```xml
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)
```

```xml
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)
```

```xml
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**

```xml
mt-audio-is-playing(3)
```

**mt-audio-is-recording**

**Description**

Returns `true()` if the client device is recording audio, `false()` otherwise.

**Usage**

```xml
mt-audio-is-recording()
```

**mt-available-db-connection-names** *(FromSolution as xs:boolean) as item()*

Returns the names of all available database (DB) connections. If `FromSolution` is set to `true()`, returns the names of DB connections in the solution. If `FromSolution` is set to `false()`, returns the names of DB connections saved on the server. The returned object is a sequence of strings.

**Examples**

- ```xml
  mt-available-db-connection-names( true() ) returns, for example, ("MyCars", "companySales", "companyContacts")
  ```
- ```xml
  mt-available-db-connection-names( false() ) returns, for example, ("DBConnOnServer-1", "DBConnOnServer-2")
  ```

**mt-available-languages**

**Description**

Returns the languages that have been defined in the Localization dialog.

In the Localization dialog, each language is specified by its ISO language code (for example: `en-US`) and a given name (for example: `English`). The language code of the default language is always the empty string, but the name can be any given string.

The function returns each language as an array of two strings; for example: `[ "en-US", "English" ]`. Multiple languages are returned as a sequence of array items; for example: `[ ["en-US", "English"], ["de-DE", "German (DE)"] ]`. The first array in the sequence will always return the default language of the design. So if no language name is specified for the default language (in the Localization dialog), then both strings of the first array item will be empty; otherwise the first array item will contain an empty string and the name that was given for the default language (for example: `[ "", "MyDefaultLanguage" ]`).

**Usage**

```xml
mt-available-languages() might return a display value of: en-US English
```

```xml
mt-available-languages() might return a display value of: en-US English de-DE German (DE)
```

```xml
mt-available-languages() might return a display value of: MyDefaultLanguage en-US English
```

**mt-base64-to-hexBinary**
**mt-base64-to-hexBinary** (Base64Image as xs:base64Binary) as xs:string

The function converts a Base64-encoded image to a hexBinary string. The Base64Image argument must be text that is encoded in base64Binary. A page source node that provides such text can be submitted.

**Usage**

mt-base64-to-hexBinary($XML1/Element1/@image) converts a Base64 image to hexBinary

**mt-cache-update-dateTime**

**Description**
Returns the time when the page-source cache was updated. If the page source is not cached then an empty sequence is returned.

**Usage**

mt-cache-update-dateTime($XML1)

**mt-called-by-enter-key**

**mt-called-by-enter-key()** as xs:boolean

Returns true() if the current action stack was triggered by pressing the Enter key, false() otherwise. Knowing that the user pressed the Enter key to start the current action stack—as opposed to, say, the Esc key—would indicate user intention, which in turn would enable conditional processing within the action stack.

**mt-called-by-escape-key**

**mt-called-by-escape-key()** as xs:boolean

Returns true() if the current action stack was triggered by the user pressing the Esc key, false() otherwise. Knowing that the user pressed the Esc key would indicate different user intention than when the Enter key were pressed, thus enabling conditional processing within the action stack.

**mt-change-image-colors** (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
XPath/XQuery Expressions and Functions

MobileTogether Extension Functions

**mt-client-ip-address**

```xml
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**

```xml
mt-connected-via-lan()
```

**mt-connected-via-wifi**

**Description**

Returns `true()` if the mobile device is connected via WiFi, `false()` otherwise.

**Usage**

```xml
mt-connected-via-wifi()
```

**mt-control-text-offset**

```xml
mt-control-text-offset(ControlKind as xs:string) as xs:integer*
mt-control-text-offset(ControlKind as xs:string, Parameters as map) as xs:integer*
```

Returns the pixel value of the left, top, right, and bottom offsets, in that order, of the content of the control named in the `ControlKind` argument. The second argument, `Parameters`, is optional and is a key-value map that defines the properties of the control. The available keys and their values are listed below. The map filters the control kind named in `ControlKind` to those instances of the control kind that match the property values listed in the map.

The return values of this function can be used to align the content of the selected controls. See example below.

Given below are the available key-value pairs that can be submitted as the map of the `Parameters` argument. The order of the key-value pairs in the map is not fixed.

- "Text Size": "small"|"medium"|"large"
- "Unit": "px"|"dp"|"sp"|"
  default is "px". For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.
- "Bold Text": "true"|"false"
- "Italic Text": "true"|"false"
- "Underline Text": "true"|"false"
- "Button Image": Any of the Button Image options (for example, "+"|"-"|">|"Share")
- "Button Background": "transparent"|"non-transparent". Default is "non-transparent".
Example

```
declare function AllOffsets($controlAsName)
{
    let $map := map{
        'Label ':' map{},
        'Edit Field: ' map{},
        'Button: ' map{},
        'Button:WithImage': map{'Button Image: 'Calendar'},
        'Button:WithImageTransparent': map{'Button Image: 'Calendar', 'Button Background' : 'transparent'},
        'Button:WithImageRight': map{'Button Image: 'Calendar', 'Button Image Position': 'right of text'},
        'Combo Box:': map{},
        'Check Box:': map{},
        'Check Box:Right': map{'Check Mark Position': 'right of text'},
        'Radio Button:': map{},
        'Radio Button:Right': map{'Check Mark Position': 'right of text'},
        'Date:': map{},
        'Time:': map{},
        'Switch:': map{
    }
    return element Root {
        for-each(map:keys($map), function($key) {
            let $offset := mt-control-text-offset(substring-before($key, ':'), $map($key))
            let $name := if ($controlAsName) then 'control' else replace($key, ': ', '')
            return element {$name} {
                attribute name {replace($key, ': ', '')},
                attribute offsets {$offset},
                attribute left {$offset[1]},
                attribute top {$offset[2]},
                attribute right {$offset[3]},
                attribute bottom {$offset[4]}
            }
        })
    }
}
```

`mt-control-width`

\[ mt-control-width(\text{Text as xs:string*}, \text{Parameters as map(*)}) \text{ as xs:integer?} \]

Returns the minimum width in pixels of the control when the `Text` string is the display-text of the control. The `Text` argument is the text that is displayed in the control. The `Parameters` argument is a key-value map that defines the properties of the control. The available keys and their values are listed below. The integer that is returned is the minimum width, in pixels, of the control when the submitted `Text` string is displayed with the properties specified in the `Parameters` argument. This value can then be used to calculate and specify other properties related to the control, such as the widths of table columns in which the control appears.
Note: This function is not available for web-client rendering. For web-client rendering, use the Measure Controls action.

Note: This function can only be used in the XPath expressions of (i) design actions and (ii) the Ensure Exists on Load (XPath Value) option of page source tree nodes. 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"|"dp"|"sp"|"": default is "px". For information about the relationships between pixels, dp (device-independent pixels), and sp (scale-independent pixels), see Sizes: Pixels, DPI, DP, SP.
- "Bold Text" : "true"|"false"
- "Italic Text" : "true"|"false"
- "Underline Text" : "true"|"false"
- "Button Image" : Any of the Button Image options (for example, "+"|"-"|">"|"Share")
- "Button Background" : "transparent"|"non-transparent". Default is "non-transparent".

Examples


mt-convert-units

mt-convert-units(Size as xs:string, TargetUnit as xs:string) as xs:string

Converts the length value specified in the Size argument to an equivalent value in the unit specified by the TargetUnit argument. You can convert between any two of the following units: px, dp, and sp. Both input arguments, as well as the output value, are strings. For more information about units and conversion between them, see Sizes: Pixels, DPI, DP, SP.

Examples

- mt-control-unit("24px", "dp") returns, depending on the device's resolution, for example, "22dp" on one device and "20dp" on another
- mt-control-unit("20sp", "px") returns, depending on the device's resolution, for example, "22px" on one device and "24px" on another

mt-db-any-changed-fields

Description

Returns true if the row element contains new, modified or deleted columns. Returns false if the fields are unmodified. The function tests whether there was any modification to the specified DB row.

Usage

mt-db-any-changed-fields($DB/DB/RowSet/Row[3])
### mt-db-any-changed-rows

**Description**
Returns **true** if the $DB variable (which represents a DB source) has new, modified or deleted rows. Returns **false** if the DB has not been modified. The function tests whether there was any modification to the DB, and works also if the RowSets element does not have a primary key.

**Usage**
```xml
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**
```xml
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**
```xml
mt-db-deleted-original-rows($DB1)
```

### mt-db-file-path

**Description**
Takes a page source (variable) as its single parameter and returns the file path of the page source's file-based database. The file path will be relative to the Solutions Working Directory.

**Note**
- This is a server-side function and can be executed on the server only.
- The function works for SQLite and Access databases.

**Usage**
```xml
mt-db-file-path($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.
MobileTogether Extension Functions

XPath/XQuery Expressions and Functions

- **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-db-original-row

**Description**

In order to enable the editing of DB rows, a copy of the original row set is saved in the DB page source in an element named OriginalRowSet, while modifications are stored in a parallel data structure named RowSet. On saving the page source, the content of RowSet is copied to OriginalRowSet. The mt-db-original-row function can be used to retrieve the original row that corresponds to the Row element (of the RowSet data structure) that is submitted as the function's single parameter. For more information, see the sections Editing DB Data and Saving Data to the DB.
Usage

mt-db-original-row($DB1/DB/RowSet/Row[1]) returns $DB1/DB/OriginalRowSet/Row[1].

mt-email-attachment

mt-email-attachment(Filename as xs:string, Content as item(), ContentType as xs:string) as array(*)

Prepares the XML, Base64, or text content that is provided by the Content argument as an email attachment. Whether the content is parsed as XML or Base64 or text is determined by the ContentType argument, which takes either XML, Base64, or text as its value. The filename that is associated with the attachment is given by the Filename argument.

Note: The mt-email-attachment is a requirement when using the Dynamic Attachments option of the Send Email To 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" </element body> </element html>

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
**XPath/XQuery Expressions and Functions**

- `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**

```xml
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**

```xml
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**

```xml
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(1234.5, '$0.0')` returns $1234.5
  - `mt-format-number(1234.55, '$0.0')` returns $1234.6

**mt-geo-map-marker**

```xml
mt-geo-map-marker(id as xs:string, geolocation as xs:string) as map (*)
mt-geo-map-marker(id as xs:string, geolocation as xs:string, popup? as (xs:string*)) as map (*)
mt-geo-map-marker(id as xs:string, geolocation as xs:string, popup? as (xs:string*), color? as xs:string) as map (*)
```

The function generates an XPath map-construct*, which is used to create a marker for the Geolocation Map control. The function's `id` and `geolocation` arguments are mandatory; the `popup` and `color` arguments are optional. Each of the submitted strings is returned as text in a value of one of the map-construct's key–value pairs (see the examples below). Each key of the returned map-construct receives its value from the corresponding argument of the function, these correspondences being determined by the index number of the argument. For example: the first argument of the function provides the value of the `id` key. Note that the function's third argument is a sequence of strings. These strings are used to generate the title and text of the marker's popup. The first string provides the popup's title; the following strings are concatenated to provide the text of the popup, with each string starting a new line. If you do not want a popup to be created, submit an empty sequence as the third argument. The `color` argument can be given as text (for example, "green") or as an RGB value (for example, "#336699"). If no `color` argument is specified, then the default marker color of the device will be used.

Note that each function returns one marker. To generate multiple markers, use a sequence of multiple `mt-geo-map-marker` functions. See examples below. Note also that you can use the `mt-geo-map-marker` function within other XPath expressions as shown in the third example below (which uses an if-then-else construction).

*XPath map-construct: An XPath datatype construction similar to an XPath array. The map-construct is a sequence of key–value pairs (see the example map that is returned in the first example of the Usage section below). Note that the semantics of “map” in “XPath map-construct” refers to a mapping of keys to
values in XPath; it does not refer to a cartographic map.

Usage

mt-geo-map-marker("vie","48.2143531 16.3707266", ("Vienna","Altova EU","European headquarters"), "green") returns a single XPath map-construct—which, in turn, generates a single marker in the Geolocation Map control:

map {
    "id":"vie",
    "geolocation":(48.2143531, 16.3707266),
    "title":"Vienna",
    "text":"Altova EU European headquarters",
    "color":"green"
}

mt-geo-map-marker("vie","48.2143531 16.3707266", ("Vienna","Altova EU")), mt-geo-map-marker("bev","42.5584577 -70.8893334", ("Beverly","Altova US")) returns two XPath map-constructs—and so two markers for the Geolocation Map control.

mt-geo-map-marker("vie","48.2143531 16.3707266", ("Vienna","Altova EU")), mt-geo-map-marker("bev","42.5584577 -70.8893334", ("Beverly","Altova US")),
if ( $XML/MapMarkers/@withLondon = "1" ) then
    mt-geo-map-marker("lon","51.50939 -0.11832", ("London","No Altova") )
else
    () returns two XPath map-constructs, plus—if the @withLondon attribute has a value of "1"—a third XPath map-construct for the London geolocation. The function in effect generates two, or three, markers for the Geolocation Map control.

▼ mt-geolocation-started

Description
Returns true() if the solution has started geolocation tracking, false() otherwise.

Usage
mt-geolocation-started()

▼ mt-get-page-source-from-name

mt-get-page-source-from-name(PageSourceName as string) as node
Returns the root element of the page source having the name that matches the submitted string. Consequently, you can access nodes of the entire document tree via XPath. If no page source on the current page has a name that matches the submitted string, then an error is thrown. The submitted string argument does not need to include the leading $ character.

Usage
mt-get-page-source-from-name("$PERSISTENT") returns the root element of the $PERSISTENT page source, for example $PERSISTENT/Orders
**mt-get-page-source-from-name**("PERSISTENT") returns the root element of the $PERSISTENT page source, for example $PERSISTENT/MyRootElement

**mt-get-page-source-name**

mt-get-page-source-name(PageSource as node()) as xs:string

Returns the name of the page source that is submitted with the PageSource argument. The name is returned as a string.

**Usage**

mt-get-page-source-name($XML1) returns the string "$XML1"
mt-get-page-source-name($PERSISTENT) returns the string "$PERSISTENT"

**mt-get-page-source-structure**

mt-get-page-source-structure(PageSource as node()) as xs:string
mt-get-page-source-structure(PageSource as node(), EnsureValues as boolean) as xs:string

Returns the structure of the page source that is submitted with the PageSource argument. If the EnsureValues argument is set to true(), then the content of nodes is also returned. If EnsureValues is set to false(), or if no second argument is supplied (see first signature above), then the page source structure is returned without content. The structure and content are those at the time the page source is loaded.

**Usage**

mt-get-page-source-structure($XML1) returns the data structure of the $XML1 page source when the page source was loaded
for $i in mt-get-page-source-structure($XML1, true()) return $i//Product[1] returns the content of the first Product element of the $XML1 page source at the time when the page source was loaded

**mt-has-serveraccess**

**Description**

Returns true if server access is possible, false otherwise. 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**

```
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 is 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 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', 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-in-app-purchase-platform-to-product**

```
mt-in-app-purchase-platform-to-product(PlatformID as xs:string) as xs:string
```

---
The argument PlatformID is the ID of a product on a given platform. It returns the product name that maps to this product ID, as defined in the In-App Purchase Products dialog. Also see the function mt-in-app-purchase-platform-to-product().

Examples

- mt-in-app-purchase-platform-to-product('AndroidID_For_MyBlogSubscription') returns, on Android devices: 'MyBlog-Subscription'
- mt-in-app-purchase-platform-to-product('AppleID_For_MyBlogSubscription') returns, on iOS devices: 'MyBlog-Subscription'
- mt-in-app-purchase-platform-to-product('WindowsID_For_MyBlogSubscription') returns, on Windows devices: 'MyBlog-Subscription'

mt-in-app-purchase-product-to-platform

mt-in-app-purchase-product-to-platform(ProductName as xs:string) as xs:string

The argument ProductName is the name of a product as defined in the In-App Purchase Products dialog. It returns the ID of that product on the current platform according to the mapping in the In-App Purchase Products dialog. Also see the function mt-in-app-purchase-platform-to-product().

Examples

- mt-in-app-purchase-product-to-platform('MyBlog-Subscription') returns, on iOS devices: 'AppleID_For_MyBlogSubscription'
- mt-in-app-purchase-product-to-platform('MyBlog-Subscription') returns, on Windows devices: 'WindowsID_For_MyBlogSubscription'

mt-in-app-purchase-service-started

mt-in-app-purchase-service-started() as xs:boolean

Checks whether the client device has all the necessary requirements to carry out in-app purchases at its respective app store, including a running billing service. For example, on Android, a user account for the app store is required to carry out an in-app purchase. The function returns true() if the service is available, false() if not.

Examples

- mt-in-app-purchase-service-started() returns true() if all requirements for in-app purchases of the relevant app store are met, false() if they are not

mt-invert-color

mt-invert-color(Color as xs:string) as xs:string

The argument Color is the RGB color code (in hexadecimal format). For example, #00FFFF. Each color component (R, G, and B) in the code is inverted, and the new color code is returned.

Examples

- mt-invert-color('#000000') returns '#FFFFFF'
- mt-invert-color('#00FFFF') returns '#FF0000'
- mt-invert-color('#AA0000') returns '#55FFFF'
- `mt-invert-color('#AA33BB')` returns '#55CC44'
- `mt-invert-color('#34A6D2')` returns '#CB592D'

### mt-is-server-purchased

**Description**

Returns `true` if all the licenses assigned to MobileTogether Server are purchased licenses, `false` if one or more assigned licenses is a trial one. Tip: If a trial license is no longer needed, unassign it.

**Note the following:**

- On clients the function returns `false` by default. Only when a request is made to a server will the purchase-state of the server's licenses be returned.
- In the simulator, the function always returns `false`. Use the corresponding simulator option to simulate a purchased license. For simulations in the designer and for trial runs on client, switching this option on simulates that MobileTogether Server licenses have been purchased. For simulations on the server, the actual purchase-state of licenses on the server is returned.

**Usage**

```
mt-is-server-purchased()
```

### mt-last-file-path

**mt-last-file-path() as xs:string?**

Returns the full file path (for example, on Android: `/storage/emulated/0/Download/MyFile.xml`) of the last client file used (loaded or saved) in any of the following actions: Audio Recording, Video Recording, Load/Save File, Load/Save Binary File, Load Image. Note that some versions of operating systems other than Android might not support this function.

**Usage**

```
mt-last-file-path()
```

### mt-last-in-app-purchase-response-code

**mt-last-in-app-purchase-response-code() as xs:integer**

Returns the success/failure code of the last in-app purchase request made by the client device at the app store. If the request was successfully executed, the code would typically be 0, otherwise it would return some other integer. The function is useful for checking whether a step of the workflow was executed correctly. Related functions are `mt-in-app-purchase-response-text()` and `mt-in-app-purchase-response-was-user-canceled()`.

**Examples**

- `mt-in-app-purchase-response-code()` typically returns 0 if the last in-app purchase request was correctly executed.

### mt-last-in-app-purchase-response-text

**mt-last-in-app-purchase-response-text() as xs:string**

Returns a text message describing the success/failure of the last in-app purchase request. The function is particularly useful if there is an error because it provides textual description of the error. Related functions are `mt-in-app-purchase-response-was-user-canceled()` and `mt-in-app-purchase-response-
Examples

- `mt-in-app-purchase-response-text()` returns a suitable text message according to the success level of the action at the app store.

mt-last-in-app-purchase-response-was-user-canceled

The function checks with the app store whether the last in-app purchase was canceled by the user. It returns `true()` if the last purchase was canceled, `false()` otherwise. Related functions are `mt-in-app-purchase-response-text()` and `mt-in-app-purchase-response-code()`. This function is useful for distinguishing this kind of failed purchase request from other kinds of error that might occur during the purchase request.

Examples

- `mt-in-app-purchase-response-was-user-canceled()` returns `true()` if the user canceled the last in-app purchase, `false()` if they are not.

mt-load-json-from-string

Description

The function takes as its argument a string that is a serialized JSON structure. The JSON structure is converted to XML and wrapped inside an element named `json`, and returned as a document node. For example, say you submit the following JSON document in a string (that is, within quotes) as the argument of the function:

```json
{
  "Root": {
    "user": "Altova",
    "message": "Hello",
    "A": {
      "B": "B Text",
      "C": "C Text",
      "D": {}
    }
  }
}
```

The function will return a document node having a document element named `json` that contains the JSON structure converted to XML. The child element of `json` will be `Root`.

Since it is a document node that is returned by the function, you can access parts of the returned document by appending an XPath locator expression to the function. For example, if you evaluate the following expression (with the locator expression highlighted in yellow):

```xml
mt-load-json-from-string('{
  "A": {
    "B": "B Text",
    "C": "C Text",
    "D": {}
  }
}
')
```
it returns "C Text".

Usage

mt-load-json-from-string('{ "A": "A Text" }') returns the document node having a document element named json, which will have one child element named A.

mt-load-json-from-string('{ "A": "A Text" }')/*/ returns all the elements of the document: that is, the json element with one child element named A.

\[ \text{mt-load-string} \]

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 the 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.

\[ \text{mt-localized-string-name} \]

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**
```xml
mt-nfc-started()
```

### mt-page-stack
**Description**
Returns a list of open pages. The list will be a sequence of strings, that is, it will consist of the names of pages with a space between each pair of names.

**Usage**
```xml
mt-page-stack()
```

### 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**
```xml
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**
```xml
mt-reload-dateTime($XML1)
```

### mt-run-appstoreapp-url
```xml
mt-run-appstoreapp-url(Scheme? as xs:string, Host? as xs:string, InputParameters? as xs:string) as xs:string?
mt-run-appstoreapp-url(InputParameters? as xs:string) as xs:string?
```
Generates the URL of a MobileTogether AppStore App from either (i) the three submitted arguments, or (ii) the single InputParameters argument. On clicking the URL, which would typically be sent in an email, the AppStore App will be started. The URL must have the format: `<url-scheme>://<url-host>`. The app's manifest file contains the scheme information, and this indicates to the device that URLs starting with this scheme should be opened by this app. For more information about AppStore Apps, see the section AppStore Apps.

- **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). If this argument is omitted or an empty string is submitted, then the scheme is set to that of the currently running app.
- **Host**: The unique host name that is associated with the app. The host is assigned when the app's program code is generated (in Screen 1 of the Generate Program Code Wizard). If this argument is omitted or an empty string is submitted, then the hostname will be that associated with the currently running app.
- **InputParameters**: Takes the function `mt-run-solution-url-parameters` as its input. The argument of the function is a sequence of string values that provide the values of the query's parameters. The `mt-run-solution-url-parameters` function returns a string containing the parameters (names and values) of the URL's query string, correctly encoded and percent-escaped according to the rules for encoding URL query strings. The parameter names in the result string are automatically generated by the function (they are: `in1, in2 ... inN`), and each is assigned a value from the string items of the function's argument, with names and values being paired in index order. (Additionally, the InputParameters argument can be provided as a string that is already encoded for the query string part of a URL (see second example below.).)

The `mt-run-appstoreapp-url` function therefore creates a URL, with or without query parameters, that opens a MobileTogether AppStore App. 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 already encoded for the query string part of a URL.

**mt-run-solution-url**

```
def mt-run-solution-url(ServerAddress? as xs:string, SolutionName? as xs:string, InputParameters? as xs:string)?
```

Generates a URL to open the specified solution in a MobileTogether client. When the URL is tapped, the Altova MobileTogether Client app is opened and the solution is started in the app. The URL is generated from either (i) the function's three submitted arguments (listed below), or (ii) the function's InputParameters argument.

- **ServerAddress**: Takes the name or IP address of the MobileTogether Server on which the solution that you want to run is deployed. If this argument is omitted or it is the empty string,
then the current server is used.

- **SolutionName**: Takes the deployed path of the solution on the server. For example: `/public/MySolution` (which would point to the `MySolution.mtd` file in the `/Public` container). If this argument is omitted or it is the empty string, then the current solution is used.

- **InputParameters**: Takes the function `mt-run-solution-url-parameters` as its input. The argument of the `mt-run-solution-url-parameters` function is either (i) a sequence of string values that will be the values of the query's parameters, or (ii) a map of `key:value` pairs that provide the name and value of the respective parameters. This function returns a string that is correctly encoded and percent-escaped according to the rules for encoding URL query strings. See the description of the `mt-run-solution-url-parameters` function below. (Additionally, the `InputParameters` argument can be provided as a string that is already encoded for the query string part of a URL (see fourth example below).)

The `mt-run-solution-url` function therefore creates a URL, with or without query parameters, that accesses a solution on a MobileTogether Server. The query parameters are passed to the solution when the solution is opened via the URL. The values of these parameters can be accessed in other design components by using the `$MT_InputParameters` global variable.

### Examples

- `mt-run-solution-url('100.00.000.1', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the server with the IP address `100.00.000.1`. The URL has no query parameters.
- `mt-run-solution-url('', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the current server. The URL has no query parameters.
- `mt-run-solution-url('', '', mt-run-solution-url-parameters(('2015', 'USA', 'true')))` returns a URL that points to the current solution on the current server. The argument of the `mt-run-solution-url-parameters` function in this example is a sequence of string values that will be the values of the query's parameters. The first string will be the value of the first parameter, the second string will be the value of the second parameter, and so on. See the description of the `mt-run-solution-url-parameters` function below.
- `mt-run-solution-url('', '', 'in1=value1&in2=value2%3FAndMoreValue2')` returns a URL that points to the current solution on the current server. The `InputParameters` argument is submitted as a string already encoded as a URL query string.

Note the following points:

- The first argument, `ServerAddress`, is used to look up information on the client about a server having the submitted name/address. The port number, user name, and user password that are associated with the server name are then used to connect to the server. So if a URL is generated with a server name that is not recognized by the client, then the URL will not work.
- The second argument, `SolutionName`: (i) generates the deployed path (on the server) if the solution is run on the server, but (ii) generates a file path for simulations.
- The third argument, `InputParameters`, uses the MobileTogether-specific XPath extension function called `mt-run-solution-url-parameters` to generate and encode the query's parameter-value pairs. Do not confuse the `mt-run-solution-url-parameters` function (which encodes the query parameters) with the `mt-run-solution-url` function (which generates the whole URL).

### mt-run-solution-url-parameters

```
mt-run-solution-url-parameters((Parameters*) as xs:string) as xs:string?
mt-run-solution-url-parameters(Map as map) as xs:string?
```
The `mt-run-solution-url-parameters` function is intended for use as the third argument of the `mt-run-solution-url` function. Its single argument is either a sequence of string values or a map of key–value pairs. The result that is generated is a single string, which will be the query string part that is to be submitted (as an argument) to the `mt-run-solution-url` function. It contains the parameters (names and values) of the query string, correctly encoded and percent-escaped according to the rules for encoding URL query strings.

**Sequence**

The argument is a sequence of string values, which are the parameter values of the query string. The parameter names in the result string are automatically generated by the function (they are: `in1`, `in2` ... `inN`), and each is assigned a value from the string items of the function's argument, with names and values being paired in index order. See the sequence examples below.

**Map**

The parameters (names and values) can also be submitted as a map of `key:value` pairs. For example: `map{"key1":"value1", "key2":"value2"}`. The ordering of parameters is not important because each parameter value is keyed to a specific parameter name. See the map examples below.

**Note:** If the `Parameters` string contains double quotes, change these to single quotes. This is required because MobileTogether uses double quotes to build the parameters string. You can use the XPath `replace` function to change double quotes to single quotes: `replace(<string>, "", ")`). Also see the `C'est la vie` examples below, where the text must be delimited by double quotes (not single quotes).

The values of these parameters can be accessed in other design components by using the `SMT_InputParameters` global variable.

**Example**

- `mt-run-solution-url-parameters(('2015', 'USA', 'true'))` returns
  "in1=2015&in2=USA&in3=true"
- `mt-run-solution-url-parameters(('2015', "USA", "true"))` returns
  "in1=2015&in2=USA&in3=true"
- `mt-run-solution-url-parameters(('2015', '', '', ''))` returns
  "in1=2015&in2=&in3=&in4="
- `mt-run-solution-url-parameters(('Stereophonics', 'C'est la vie'))` returns
  "in1=Stereophonics&in2=C#est%20la%20vie"
- `mt-run-solution-url-parameters(map('Year':'2015', 'Country':'USA', 'Include':'true'))` returns
  "Country=USA&Include=true&Year=2015"
- `mt-run-solution-url-parameters(map('Artist':'Stereophonics', 'Title':'C'est la vie'))` returns
  "Country=USA&Include=true&Year=2015"

**mt-run-web-url**

```
mt-run-web-url(ServerAddress? as xs:string, SolutionName? as xs:string, InputParameters? as xs:string) as xs:string?
mt-run-web-url(InputParameters? as xs:string) as xs:string?
```

Generates a URL, which is entered in a web browser, to open the specified solution in the browser. The URL is generated from either (i) the function's three submitted arguments (*listed below*), or (ii) the function's `InputParameters` argument:
XPath/XQuery Expressions and Functions

MobileTogether Extension Functions

- **ServerAddress**: Takes the name or IP address of the MobileTogether Server on which the solution that you want to run is deployed. If this argument is omitted or it is the empty string, then the current server is used.

- **SolutionName**: Takes the deployed path of the solution on the server. For example: `/public/MySolution` (which would point to the `MySolution.mtd` file in the `/Public` container). If this argument is omitted or it is the empty string, then the current solution is used.

- **InputParameters**: Takes the function `mt-run-solution-url-parameters` as its input. The argument of this function is a sequence of string values that provide the values of the query’s parameters. The `mt-run-solution-url-parameters` function returns a string containing the parameters (names and values) of the URL’s query string, correctly encoded and percent-escaped according to the rules for encoding URL query strings. The parameter names in the result string are automatically generated by the function (they are: `in1`, `in2` ... `inN`), and each is assigned a value from the string items of the function’s argument, with names and values being paired in index order. (Alternatively, the InputParameters argument can be provided as a string that is already encoded for the query string part of a URL (see fourth example below.).)

The `mt-run-web-url` function therefore creates a URL, with or without query parameters, that accesses a solution on a MobileTogether Server. The query parameters are passed to the solution when the solution is opened via the URL. The values of these query parameters can be accessed in other design components by using the `$MT_InputParameters` global variable.

**Examples**

- `mt-run-web-url('100.00.000.1', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the server with the IP address 100.00.000.1. The URL has no query parameters.

- `mt-run-web-url('', '/public/MyDesign', '')` returns a URL that points to the `MyDesign` solution on the current server. The URL has no query parameters.

- `mt-run-web-url('', '', `mt-run-solution-url-parameters(('2015', 'USA', 'true'))`)` returns a URL that points to the current solution on the current server. The argument of the `mt-run-solution-url-parameters` function is a sequence of string values that will be the values of the query’s parameters. The first string will be the value of the first parameter, the second string will be the value of the second parameter, and so on. The `mt-run-solution-url-parameters` function returns a string that is correctly encoded and percent-escaped according to the rules for encoding URL query strings.

- `mt-run-web-url('', '', 'in1=value1&in2=value2%3FAndMoreValue2')` returns a URL that points to the current solution on the current server. The InputParameters argument is submitted as a string already encoded as a URL query string.

Note the following points:

- The first argument, ServerAddress, is used to look up information on the client about a server having the submitted name/address. The port number, user name, and user password that are associated with the server name are then used to connect to the server. So, if a URL is generated with a server name that is not recognized by the client, then the URL will not work.

- The second argument, SolutionName: (i) generates the deployed path (on the server) if the solution is run on the server, but (ii) generates a file path for simulations.

- The third argument, InputParameters, uses the MobileTogether-specific XPath extension function called `mt-run-solution-url-parameters` to generate and encode the query’s parameter-value pairs. Do not confuse the `mt-run-solution-url-parameters` function (which encodes the query
XPath/XQuery Expressions and Functions

parameters) with the `mt-run-solution-url` function (which generates the whole URL).

▼ mt-save-json-to-string

**Description**
Takes a JSON node as its single argument, and returns the contents of the node as a string that is serialized in JSON format. The submitted JSON node could be an entire JSON document, or a part of a JSON document. For example, say you have the following JSON document associated with a JSON page source named `$JSON1`:

```json
{
    "Root": {
        "user": "Altova",
        "message": "Hello",
        "A": {
            "B": "B Text",
            "C": "C Text",
            "D": {}
        }
    }
}
```

If the document node is submitted (that is, `$JSON1`, which is the virtual node that is the document's parent node), then the contents of the entire document object is returned as a string. If the `$JSON1/json/Root` node is submitted as the argument, then the object that is the value of the "Root" key is returned as a string. Also see the examples below.

**Usage**
- `mt-save-json-to-string($JSON/json/Root/A/B)` returns "B Text"
- `mt-save-json-to-string($JSON/json/Root/A/D)` returns '{}'

▼ mt-server-config-url

`mt-server-config-url(ServerSettings as map) as xs:string?`

The `mt-server-config-url` function takes a map as its argument and returns a string that is a URL. When the URL is sent as a link to client devices, and the link is tapped, then server settings on the client are automatically updated. The URL will look something like this: `mobiletogether://mt/change-settings?settings=<json encoded settings>`

The JSON-encoded server settings that are contained in the URL are provided by the `ServerSettings` argument of the `mt-server-config-url` function. The `ServerSettings` map is shown below. For an example of how to use this function, open and test the example file `ClientConfiguration.mtd` in the `MobileTogetherExamples/SimpleApps` folder.

```xml
<ServerSettings>
    <DelOthSrv>false()</DelOthSrv>
    <DetView>true()</DetView>
    <Refresh>true()</Refresh>
    <RetToSln>true())</RetToSln>
</ServerSettings>
```

© 2015-2021 Altova GmbH
"ActSrvURL": "",  (: the first server with this URL gets the active one :) 
"Servers": array{ 
  map{ 
    "Name": "", 
    "URL": "",  (: if DelOthSrv is false then this property is used as key to merge the new settings with the existing ones :) 
  "LoginProvider": map{ 
    "NameSuffix": "", 
    "NamePrefix": "", 
  }, 
  "Port": ", 
  "User": ", 
  "StorePW": true(), 
  "Password": ", 
  "SSL": false() 
}, 
}, (:, map {...} to add another server :) }

mt-server-variable

mt-server-variable(VariableName as xs:string) as xs:string
Returns the value of the server variable named in the VariableName argument. Server variables are variables that are stored in the mobiletogetherserver.cfg file. In this file, the server variables are stored in the [Server Variables] section, as shown in the snippet below:

[Server Variables]
Environment=Admin
Manual=AdminDocs
StartPage=Admin

Note: The mt-server-variable function is a server-side function and can be executed on the MobileTogether Server only.

The mobiletogetherserver.cfg file is located by default in the application data folder, the location of which is given below for each operating system. You can edit the .cfg configuration file in a text editor.

Linux /var/opt/Altova/ExeNameUpperLower
Mac /var/Altova/ExeNameUpperLower
Windows C:\ProgramData\Altova\ExeNameUpperLower

For information about how to simulate server variables, see the Options tab Simulation 2.

Examples

• mt-control-unit("Environment") returns, for example, "Development" or "Portal"

mt-solution-path

mt-solution-path() as xs:string
Returns the path of the currently running solution.

mt-string-to-hexBinary

```xml
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 is 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-table-rowgroup-count

```xml
mt-table-rowgroup-count(VisibleOnly? as xs:boolean) as xs:integer
```

The function must be set inside a row-group, which becomes the context of the function. The function returns the number (count) of row-groups in the table within which the context row-group is located. The VisibleOnly argument is optional; it can have a value of true() or false(). If set to true(), then the function returns the count of the visible row-groups in the table; if set to false(), then the function returns the count of all row-groups (visible and invisible). (Visible row-groups are row-groups that have their Visible property set to true; see here for a description of table properties.) If the optional VisibleOnly argument is not submitted, then the count of visible row-groups is returned; the effect is the same as if submitting the VisibleOnly argument with a value of true().

**Note:** If the Visible property of a row-group has been defined, then a child row-group cannot use the mt-table-rowgroup-count function for visible row-groups.

**Examples**

- `mt-table-rowgroup-count()` returns 10 if the total number of row-groups in the current table is 10, all visible
- `mt-table-rowgroup-count()` returns 7 if, out of a total of 10 row-groups in the current table, seven are visible and three are invisible
- `mt-table-rowgroup-count(true())` returns 7 if, out of a total of 10 row-groups in the current table, seven are visible and three are invisible
- `mt-table-rowgroup-count(false())` returns 10 if, out of a total of 10 row-groups in the current table, seven are visible and three are invisible

mt-table-rowgroup-index

```xml
mt-table-rowgroup-index(VisibleOnly? as xs:boolean) as xs:integer
```

The function must be set inside a row-group, which becomes the context of the function. The function returns the position (or index) of the current row-group within the number of row-groups of the current table. If the optional VisibleOnly argument is set to true(), then the function returns the index of the current row-group within the set of visible row-groups of the table; if set to false(), then the index among all row-groups (visible and invisible) is returned. (Visible row-groups are row-groups that have their Visible property set to true; see here for a description of table properties.) If the optional VisibleOnly argument is not submitted, then the index among visible row-groups is returned.
• **Examples**
  - `mt-table-rowgroup-index()` returns 1 if the current row-group is the first of 10 row-groups in the current table, all row-groups visible
  - `mt-table-rowgroup-index()` returns 1 if the current row-group is the second row-group and the first row-group is invisible
  - `mt-table-rowgroup-index(true())` returns 1 if the current row-group is the second row-group and the first row-group is invisible
  - `mt-table-rowgroup-index(false())` returns 2 if the current row-group is the second row-group and the first row-group is invisible

**Examples**

- `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":""}

**Examples**

- `mt-text-to-speech-is-language-available()` as xs:boolean
  The `Language` argument can take string values of the form `en` (language code) or `en-US` (language-country code). If the language specified in the `Language` argument is available on the mobile device, then the function returns `true()`, otherwise `false()`.

**Usage**

- `mt-text-to-speech-is-language-available("en")` returns `true()` if `en` or any `en-<country>` language variant is available on the mobile device, `false()` otherwise.
- `mt-text-to-speech-is-language-available("en-US")` returns `true()` if `en-us` is available on the mobile device, `false()` otherwise.

**Examples**

- `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.

**Examples**

- `mt-transform-image()` as xs:boolean
  See the description of this function in the Image-related Functions section of the Altova Extension Functions.

**Examples**

- `mt-user-tried-to-cancel-actions()` as xs:boolean
  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()
```
12.3 User-Defined XPath/XQuery Functions

You can create your own custom-made XPath/XQuery functions for individual projects, which you can then use in all XPath expressions in the project. The access point for creating and managing these user-defined functions is the XPath Functions dialog, accessed with the command, `Project | XPath/XQuery Functions`. 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).

![New XPath Function dialog](image)

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.
12.4 FAQ about XPath/XQuery

▼ I have an XPath expression inside the repeating row of a table. Should I use an absolute or relative XPath expression to target a child attribute?

If your XPath expression is inside the repeating row of a table, then the element corresponding to the row of the table is the context node, say, Row.

- If you use an absolute path, for example $XML/Row/@id, then the XPath expression will return a sequence of the @id values of all Row elements. If you are using an operation that expects one atomic value, the operation will generate an error.
- If you use a relative path, for example @id, then, since for every repeating row, you have a context $XML/Row, the XPath expression will correctly return the single atomic value of the one @id attribute of the current Row element.

▼ I have an XPath expression that locates a mixed-content element (text and element). Instead of getting the text value of the located element and its descendants (as mandated by XPath), I get the text content of the located element only. Why?

If an element with mixed content (text and element/s) is located with an XPath locator expression, then the text content of the mixed-content element only is returned. The text content of descendant elements is ignored.

This is best explained with an example of the Update Node(s) action. Consider the Update Node(s) action defined in the screenshot below.

If the XML tree had the following structure and content:

```xml
<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:

```xml
<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($XML/Element1/source, '')` will return "AAAABBB".

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.
13 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.
Global Variables

- **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')`
13.1 Static Global Variables

Static-value variables are called *Static 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()*.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_Android</td>
<td>true()</td>
</tr>
<tr>
<td>MT_Browser</td>
<td>true()</td>
</tr>
<tr>
<td>MT_iOS</td>
<td>true()</td>
</tr>
<tr>
<td>MT_iPad</td>
<td>true()</td>
</tr>
<tr>
<td>MT_Windows</td>
<td>true()</td>
</tr>
<tr>
<td>MT_WindowsPhone</td>
<td>true()</td>
</tr>
</tbody>
</table>

### 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()*.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_SMSAvailable</td>
<td>true()</td>
</tr>
<tr>
<td>MT_TelephonyAvailable</td>
<td>true()</td>
</tr>
</tbody>
</table>

### 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-takings, geolocation-related actions.
, 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_CameraAvailable</td>
<td>true()</td>
</tr>
<tr>
<td>MT_GeolocationAvailable</td>
<td>true()</td>
</tr>
<tr>
<td>MT_NFCAvailable</td>
<td>true()</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_DeviceHeight</td>
<td>Length value in pixels</td>
</tr>
<tr>
<td>MT_DeviceWidth</td>
<td>Length value in pixels</td>
</tr>
<tr>
<td>MT_DPIX</td>
<td>Horizontal pixel density in pixels per inch</td>
</tr>
<tr>
<td>MT_DPIY</td>
<td>Vertical pixel density in pixels per inch</td>
</tr>
</tbody>
</table>

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 #ffaff.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_ButtonBackgroundColor</td>
<td>Background color of buttons; Hex values, e.g: #ffaff</td>
</tr>
<tr>
<td>MT_ButtonTextColor</td>
<td>Text color of buttons; Hex values, e.g: #336699</td>
</tr>
<tr>
<td>MT_EditFieldBackgroundColor</td>
<td>Background color of edit fields; Hex values, e.g: #ffaff</td>
</tr>
<tr>
<td>MT_EditFieldTextColor</td>
<td>Text color of edit fields; Hex values, e.g: #336699</td>
</tr>
<tr>
<td>MT_LabelBackgroundColor</td>
<td>Background color of labels; Hex values, e.g: #ffaff</td>
</tr>
<tr>
<td>MT_LabelTextColor</td>
<td>Text color of labels; Hex values, e.g: #336699</td>
</tr>
<tr>
<td>MT_PageBackgroundColor</td>
<td>Background color of pages; Hex values, e.g: #ffaff</td>
</tr>
</tbody>
</table>

Miscellaneous

- MT_AuthenticationToken
The **Solution Execution** provides a setting for passing an authentication token to a web page when the solution is executed on web clients. This variable holds the authentication data sent by the action.

- **MT_ClientLanguage**
  The language on the client device.

- **MT_InputParameters**
  Parameter values are passed to the solution when the solution is started. These values are stored in the **MT_InputParameters** variable. By default the data structure that is stored in the variable is a map (for example: \{"name":"Altova", "location":"Boston"\}). You can change the data structure of this variable in individual projects (in the More Project Settings dialog) to be a sequence of values (for example: \{"Altova", "Boston"\}).

  Currently, parameter values are passed to the solution when a hyperlink to the solution is clicked. 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 either (i) as a map, or (ii) as a sequence of string-value items that are alphabetically sorted on the query keys (for example: if the query has three keys, these are sorted alphabetically and their respective values will be the corresponding items, by index position, in the **MT_InputParameters** sequence). To retrieve a single parameter value from the sequence of values, you must know that parameter's index position in the sequence. You can then use this position in an XPath locator expression, for example: $\text{MT-InputParameters}[1]$. 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 App Store 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-sequence value indicates that the solution is running in an actual end-user scenario, and not in a simulation. $\text{MT_SimulationMode}$ 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.

<table>
<thead>
<tr>
<th>Simulation Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;designer&quot;</td>
<td>Simulation runs directly in designer</td>
</tr>
<tr>
<td>&quot;designer-server&quot;</td>
<td>Simulation with a standalone server</td>
</tr>
<tr>
<td>&quot;designer-client&quot;</td>
<td>Simulation is a trial run on client</td>
</tr>
<tr>
<td>()</td>
<td>Server to client/browser, run by end user</td>
</tr>
</tbody>
</table>
MT_UserName

The name with which to log in to MobileTogether Server.
13.2 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MT_Portrait</td>
<td>true()</td>
</tr>
<tr>
<td>$MT_Landscape</td>
<td>true()</td>
</tr>
</tbody>
</table>

- **Variables that indicate the device's viewport dimensions**
  
  **Description**
  These variables provide, respectively, the width (X dimension) and height (Y dimension) of the device's viewport. Note that the value of the X dimension changes with the orientation (portrait/landscape), as does the Y dimension. The viewport is the screen area on which design components are drawn; it is the screen area minus the top and/or bottom bars that hold tabs/buttons. In web-browser clients, the $MT_CanvasX and $MT_CanvasY variables give the dimensions of the canvas on which the MobileTogether Client app is displayed (that is, the dimensions of the browser window minus the title bar, ribbon, status bar, and any sidebars). The values of these variables are pixel values and will necessarily be less than the device's height and width dimensions (returned respectively by $MT_DeviceHeight and $MT_DeviceWidth). See the note Points Versus Pixels on iOS devices below.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MT_CanvasX</td>
<td>Width, as a length value in pixels</td>
</tr>
<tr>
<td>$MT_CanvasY</td>
<td>Height, as a length value in pixels</td>
</tr>
</tbody>
</table>

- **Points versus pixels on iOS devices**
  
  If you enter pixel values for properties that define lengths, then, when rendering for iOS devices, these values are read as *points* in the viewport coordinate space. The *viewport coordinate space* is the canvas on which the design components are drawn, and a *point* is the length unit in this space; a point here is not the typographical unit that is equal to 1/72 of an inch. The iOS device automatically maps the *points* of the viewport coordinate space to *pixels* in the device coordinate space. Mapping the values in this way (from viewport values to device values) ensures that design components
Global Variables

Dynamic Local Variables

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

In MobileTogether Designer, you can use the \$MT_CanvasX and \$MT_CanvasY dynamic variables to find out the current viewport (canvas) dimensions and so obtain lengths that are relative to these dimensions. (For iOS devices, the values returned by these variables are calculated as follows: The pixel dimensions of the current device coordinate space are converted (using an appropriate conversion factor) to point values in the viewport coordinate space. These point values—as numbers—are returned by the variables as pixels for use in the design.) For example, if you want an image to be half the width of the viewport, then give it a width in pixels that is equal to \$MT_CanvasX * 0.5; the XPath expression for this image width would be \concat(\$MT_CanvasX * 0.5, 'px').

- Variables that contain the size of resizable windows (in Windows apps and browsers)

Description
These variables are applicable only for 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).

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_WindowHeight</td>
<td>A length value in pixels</td>
</tr>
<tr>
<td>MT_WindowWidth</td>
<td>A length value in pixels</td>
</tr>
</tbody>
</table>

- Variables that contain information about the current control

Description
These variables contain information related to the current control and its associated page source node (the source node 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: \textit{if} \{\$MT_ControlValue = 'NaN'} \textit{then} '#FF0000' \textit{else} '#FFFFFF'.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_ControlKind</td>
<td>The kind of control, as a string. Example: &quot;Label&quot;</td>
</tr>
</tbody>
</table>
**Global Variables**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MT_ControlName</td>
<td>The name you give the control, as string. Example: &quot;Label-1&quot;</td>
</tr>
<tr>
<td>MT_ControlNode</td>
<td>XML node that is the control's source node</td>
</tr>
<tr>
<td>MT_ControlValue</td>
<td>Value of the control's page-source-link node</td>
</tr>
<tr>
<td>MT_ControlValueBeforeChange</td>
<td>Previous value of the control's page-source-link node, before the control or node is edited</td>
</tr>
</tbody>
</table>

**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_GeolocationMapMarker**
  
  The dynamic variable `$MT_GeolocationMapMarker` contains information about the marker that was last clicked by the client's user. This information that is contained in the variable is stored in an XPath-map construct, in a format that is shown by the following example:

  ```
  map {
    "id":"vie",
    "geolocation":(48.2143531, 16.3707266),
    "title":"Vienna",
    "text":"Altova EU"
  }
  ```

  To fetch a value from the XPath-map construct, use an XPath expression like this: `map:get( $MT_GeolocationMapMarker, "id" )`. This particular expression returns the value of the `id` key (that is, the `id` of the marker that was clicked).

- **MT_HTTPExecute_Result**
  
  The XML result of the last executed Execute SOAP Request or Execute REST Request.

- **MT_MeasureControls**
Automatically stores the result of the last-executed Measure Controls action.

- **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_UpdatedInAppPurchases**
  Contains a sequence of SKU-IDs, as strings, of in-app purchases that have been updated in the In-App-Purchase Page Source. The variable is in scope only while actions of the OnPurchaseUpdated event are being processed.

- **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.
13.3 User Variables

User Variables are variables that you define in the lower pane of the Global Variables dialog (Project | Global Variables, screenshot below). They are very useful if you wish to store data that can be accessed by multiple objects at different times during execution.

To add a user variable, in the lower pane, do the following:

1. Click the Append or Insert icons (located in the pane's toolbar) to add a new entry to the list.
2. Enter the name of your new variable (in the Name column, without a $ symbol) and give the variable a description (Description column). See screenshot above.
3. Click in the Value field to bring up the Edit XPath/XQuery Expression dialog, and enter the XPath expression that determines the value of the variable.
4. If you want the variable to be stored on the client only, select the Client Only option. This option is useful if the variable contains or involves a large dataset and you want to avoid possible performance slowdowns that might occur when data is transferred between client and server.
5. Select an icon to help identify the new variable as belonging to a particular group.
6. Click OK to finish. The variable is added as a global variable, and can be used in programming contexts.
Variables to store styling property values

You can store the value of a styling property in a variable: as a string that exactly matches a valid CSS property value. For example, you could set a variable named MyTextColor and give it a value of "#AA6633". You could then use the variable $MyTextColor as the property of Text Color wherever desired in the design.

A set of commonly used properties can be specified via variables, and the styles they define will be displayed not only at run time and during simulations, but also in the design itself.

This feature (styling via variables) is currently provided for the following style properties:

- All color styles (text, background, etc)
- Line styles (color, style, width)
- Text styles (size, weight, etc)
- All paddings and margins
- Borders
- Vertical and horizontal alignment
- Table column width, maximum column width
- Control width and height

Note the following points:

- Where units need to be specified, write the units correctly. For example, "16px". If you have any doubt about the correct unit to use or about the way to write the unit, look up that property's available units in the Styles & Properties Pane.
- The only supported length unit is pixels (px).
- The value of the variable can only be a literal value, expressed as a single string.
- If you use an XPath expression that uses operators, constructs, or functions (for example, an if...then...else construct), then the variable that uses such a construct cannot be used, but it can be passed to a styling variable. The styling variable will work if the first variable has been evaluated already (typically in the static context) and if it contains a string when it is passed to the styling variable.
- The XPath expressions of user-defined variables are re-evaluated only in situations where global variable values can change, for example, when the orientation of the client device is changed.
14 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. 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.
<table>
<thead>
<tr>
<th>Database</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firebird 2.5.4, 3.0</td>
<td></td>
</tr>
<tr>
<td>IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5, 11.5</td>
<td></td>
</tr>
<tr>
<td>IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4</td>
<td>Logical files are supported and shown as views.</td>
</tr>
<tr>
<td>IBM Informix 11.70, 12.10, 14.10</td>
<td></td>
</tr>
<tr>
<td>MariaDB 10.2, 10.3, 10.4, 10.5</td>
<td></td>
</tr>
<tr>
<td>Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019</td>
<td>At the time of writing (early September 2019), there is no Microsoft Access Runtime available for Access 2019. You can connect to an Access 2019 database from Altova products only if Microsoft Access 2016 Runtime is installed and only if the database does not use the &quot;Large Number&quot; data type.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>SQL Server 2016 codebase</td>
</tr>
<tr>
<td>MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0</td>
<td></td>
</tr>
<tr>
<td>Oracle 9i, 10g, 11g, 12c, 18, 19</td>
<td></td>
</tr>
<tr>
<td>PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0, 9.1, 9.2, 9.4, 9.5, 9.6, 10, 11, 12.1, 13</td>
<td>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.</td>
</tr>
<tr>
<td>Progress OpenEdge 11.6</td>
<td></td>
</tr>
<tr>
<td>SQLite 3.33.0</td>
<td>SQLite connections are supported as native, direct connections to the SQLite database file. No separate drivers are required.</td>
</tr>
<tr>
<td>Sybase ASE 15, 16</td>
<td></td>
</tr>
<tr>
<td>Teradata 16</td>
<td></td>
</tr>
</tbody>
</table>

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.
14.1  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 as page sources to the design of a page and then used within it. Whether a DB page source is editable or not is defined at the time the page source is added. Specify a DB page source to be uneditable if its data is required only for presentation. Make the page source editable if you want to allow clients to modify DB data.

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 page source is used on multiple pages, then a single tree structure can be shared across all instances of the page source. The option to share a tree structure is available each time a page source is added that is used on another page. When a shared structure is modified, you are offered the option of modifying the shared page source in its multiple instances across pages; alternatively, the shared page source is modified only in the instance in which it is modified.
Switching page sources

After you have created a design that uses a DB page source, you can switch the page source to another DB that has the same structure and continue to use the original design. To switch the DBs of a page source, right-click the $DB root node of the tree, select Choose DB Data Source, and continue with the DB connection process.

Two DBs are considered to have the same structure if they have the same table names, same column names, and same column definitions. If the new structure is different in any way, although the connection to the DB will be made, the page source will not be updated with data from the new DB. If the DB switch is aborted, then the page source will continue to use the original DB.

Note: If the DBs involved in the switch have different case-sensitivities, then you will have to modify SQL statements, XPath expressions, and any other constructs that use the non-matching names.

Tree structure of the DB page source

Every DB page source has the following structure:

```
$DBX (the root node)
  |
  --DB (the root element)
    |
    |--RowSet (a container element for the rows of the DB table)
      |
      |--Row (the rows of the DB table)
        |
        |--<Attributes> (the columns of the DB table)
```
The nodes in the tree can be addressed using XPath expressions. If a node is set as the XPath context node for the page (via the node's context menu), then XPath expressions can be built relative to the XPath context node. Otherwise nodes can be addressed using absolute paths starting at the root node: $DBX/DB/RowSet/Row/MyAttribute.

You can also use XQuery expressions to retrieve or manipulate data in the DB tree. See the section on primary keys below for an example.

About OriginalRowSet

In order for data to be edited and saved, the tree of the page source must also include an OriginalRowSet element, which is a copy of the RowSet element. The original data is saved in the OriginalRowSet element, while edited data is saved in the RowSet element. When the page source is saved, the difference between the two trees, OriginalRowSet and RowSet, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to OriginalRowSet so that OriginalRowSet contains the newly saved DB data, and the modification process can be repeated.

Note the following points:

- The OriginalRowSet element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command Create OriginalRowSet.
- The Create OriginalRowSet command is enabled for database type ($DB) root nodes. It is a toggle command that creates/removes an OriginalRowSet data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the OriginalRowSet structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function mt-db-original-row.

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:

```xquery
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
```

![XQuery example](image)
14.2 Connecting to a Data Source

In the most simple case, a database can be a local file such as a Microsoft Access or SQLite database file. In a more advanced scenario, a database may reside on a remote or network database server which does not necessarily use the same operating system as the application that connects to it and consumes data. For example, while MobileTogether Designer runs on a Windows operating system, the database from which you want to access data (for example, MySQL) might run on a Linux machine.

To interact with various database types, both remote and local, MobileTogether Designer relies on the data connection interfaces and database drivers that are already available on your operating system or released periodically by the major database vendors. In the constantly evolving landscape of database technologies, this approach caters for better cross-platform flexibility and interoperability.

The 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
Databases

Connecting to a Data Source

- 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.

14.2.1 Starting the Connection Wizard

MobileTogether Designer provides a wizard that guides you through the steps required to set up a connection to a data source.

Before you go through the wizard steps, be aware that for some database types it is necessary to install and 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.
Connecting to a Data Source

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

14.2.2 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.
<table>
<thead>
<tr>
<th>Database</th>
<th>Interface</th>
<th>Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firebird</td>
<td>ADO.NET</td>
<td>Firebird ADO.NET Data Provider (<a href="https://www.firebirdsql.org/en/additional-downloads/">https://www.firebirdsql.org/en/additional-downloads/</a>)</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>Firebird JDBC driver (<a href="https://www.firebirdsql.org/en/jdbc-driver/">https://www.firebirdsql.org/en/jdbc-driver/</a>)</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>Firebird ODBC driver (<a href="https://www.firebirdsql.org/en/odbc-driver/">https://www.firebirdsql.org/en/odbc-driver/</a>)</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>ADO</td>
<td>IBM OLE DB Provider for DB2</td>
</tr>
<tr>
<td></td>
<td>ADO.NET</td>
<td>IBM Data Server Provider for .NET</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>IBM Data Server Driver for JDBC and SQLJ</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>IBM DB2 ODBC Driver</td>
</tr>
<tr>
<td>IBM DB2 for i</td>
<td>ADO</td>
<td>IBM DB2 for i5/OS IBMDA400 OLE DB Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM DB2 for i5/OS IBMDARLA OLE DB Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBM DB2 for i5/OS IBMDASQL OLE DB Provider</td>
</tr>
<tr>
<td></td>
<td>ADO.NET</td>
<td>.NET Framework Data Provider for IBM i</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>IBM Toolbox for Java JDBC Driver</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>iSeries Access ODBC Driver</td>
</tr>
<tr>
<td>IBM Informix</td>
<td>ADO</td>
<td>IBM Informix OLE DB Provider</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>IBM Informix JDBC Driver</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>IBM Informix ODBC Driver</td>
</tr>
<tr>
<td>Microsoft Access</td>
<td>ADO</td>
<td>Microsoft Jet OLE DB Provider</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft Access Database Engine OLE DB Provider</td>
</tr>
<tr>
<td></td>
<td>ADO.NET</td>
<td>.NET Framework Data Provider for OLE DB</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>Microsoft Access Driver</td>
</tr>
<tr>
<td>MariaDB</td>
<td>ADO.NET</td>
<td>In the absence of a dedicated .NET connector for MariaDB, use Connector/NET for MySQL (<a href="https://dev.mysql.com/downloads/connector/net/">https://dev.mysql.com/downloads/connector/net/</a>).</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>MariaDB Connector/J (<a href="https://downloads.mariadb.org/">https://downloads.mariadb.org/</a>)</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>MariaDB Connector/ODBC (<a href="https://downloads.mariadb.org/">https://downloads.mariadb.org/</a>)</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>ADO</td>
<td>Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Microsoft OLE DB Provider for SQL Server (SQLOLEDB)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SQL Server Native Client (SQLNCLI)</td>
</tr>
<tr>
<td></td>
<td>ADO.NET</td>
<td>.NET Framework Data Provider for SQL Server</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.NET Framework Data Provider for OLE DB</td>
</tr>
<tr>
<td>Database</td>
<td>Interface</td>
<td>Drivers</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>JDBC</td>
<td>Microsoft JDBC Driver for SQL Server (<a href="https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server">https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server</a>)</td>
<td></td>
</tr>
<tr>
<td>ODBC</td>
<td>ODBC Driver for Microsoft SQL Server (<a href="https://docs.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server">https://docs.microsoft.com/en-us/sql/connect/odbc/download-odbc-driver-for-sql-server</a>)</td>
<td></td>
</tr>
<tr>
<td>MySQL</td>
<td>ADO.NET</td>
<td>Connector/NET (<a href="https://dev.mysql.com/downloads/connector/net/">https://dev.mysql.com/downloads/connector/net/</a>)</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>Connector/J (<a href="https://dev.mysql.com/downloads/connector/j/">https://dev.mysql.com/downloads/connector/j/</a>)</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>Connector/ODBC (<a href="https://dev.mysql.com/downloads/connector/odbc/">https://dev.mysql.com/downloads/connector/odbc/</a>)</td>
</tr>
<tr>
<td>Oracle</td>
<td>ADO</td>
<td>Oracle Provider for OLE DB</td>
</tr>
<tr>
<td></td>
<td>ADO.NET</td>
<td>Microsoft OLE DB Provider for Oracle</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>Oracle Data Provider for .NET (<a href="http://www.oracle.com/technetwork/topics/dotnet/index-085163.html">http://www.oracle.com/technetwork/topics/dotnet/index-085163.html</a>)</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>Microsoft ODBC for Oracle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Oracle ODBC Driver (typically installed during the installation of your Oracle database client)</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>JDBC</td>
<td>PostgreSQL JDBC Driver (<a href="https://jdbc.postgresql.org/download.html">https://jdbc.postgresql.org/download.html</a>)</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>psqlODBC (<a href="https://odbc.postgresql.org/">https://odbc.postgresql.org/</a>)</td>
</tr>
<tr>
<td></td>
<td>Native Connection</td>
<td>Available. There is no need to install any drivers if using native connection.</td>
</tr>
<tr>
<td>Progress OpenEdge</td>
<td>JDBC</td>
<td>JDBC Connector (<a href="https://www.progress.com/jdbc/openedge">https://www.progress.com/jdbc/openedge</a>)</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>ODBC Connector (<a href="https://www.progress.com/odbc/openedge">https://www.progress.com/odbc/openedge</a>)</td>
</tr>
<tr>
<td>SQLite</td>
<td>Native Connection</td>
<td>Available. There is no need to install any drivers if using native connection.</td>
</tr>
<tr>
<td>Sybase</td>
<td>ADO</td>
<td>Sybase ASE OLE DB Provider</td>
</tr>
<tr>
<td></td>
<td>JDBC</td>
<td>jConnect™ for JDBC</td>
</tr>
<tr>
<td></td>
<td>ODBC</td>
<td>Sybase ASE ODBC Driver</td>
</tr>
</tbody>
</table>
## 14.2.3 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 a precursor of the newer ADO.NET and is still one of the possible ways to connect to Microsoft native databases such as Microsoft Access or SQL Server, although you can also use it for other data sources.

Importantly, you can choose between multiple ADO providers, and some of them must be downloaded and installed on your workstation before you can use them. For example, for connecting to SQL Server, the following ADO providers are available:

- Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL)
- Microsoft OLE DB Provider for SQL Server (SQLOLEDB)
- SQL Server Native Client (SQLNCLI)

From the providers listed above, the recommended one is MSOLEDBSQL; you can download it from [https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15](https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15). Note that it must match the platform of MobileTogether Designer (32-bit or 64-bit). The SQLOLEDB and SQLNCLI providers are considered deprecated and thus are not recommended.

The **Microsoft OLE DB Provider for SQL Server (SQLOLEDB)** is known to have issues with parameter binding of complex queries like Common Table Expressions (CTE) and nested SELECT statements.

To set up an ADO connection:

1. Start the database connection wizard.
2. Click ADO Connections.
3. Click **Build**.

![ADO Connections](image)

4. Select the data provider through which you want to connect. The table below lists a few common scenarios.

<table>
<thead>
<tr>
<th>Provider</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLE DB Provider(s)</td>
</tr>
<tr>
<td>Microsoft Jet 4.0 OLE DB Provider</td>
</tr>
<tr>
<td>Microsoft Office 12.0 Access Database Engine OLE DB Pro</td>
</tr>
<tr>
<td>Microsoft Office 15.0 Access Database Engine OLE DB Pro</td>
</tr>
<tr>
<td>Microsoft OLE DB Provider for Analysis Services 11.0</td>
</tr>
<tr>
<td>Microsoft OLE DB Provider for ODBC Drivers</td>
</tr>
<tr>
<td>Microsoft OLE DB Provider for Oracle</td>
</tr>
<tr>
<td>Microsoft OLE DB Provider for Search</td>
</tr>
<tr>
<td>Microsoft OLE DB Provider for SQL Server</td>
</tr>
<tr>
<td>Microsoft OLE DB Simple Provider</td>
</tr>
<tr>
<td>MSDataShape</td>
</tr>
<tr>
<td>OLE DB Provider for Microsoft Directory Services</td>
</tr>
<tr>
<td>SQL Server Native Client 10.0</td>
</tr>
<tr>
<td>VSEE Versioning Enlistment Manager Proxy Data Source</td>
</tr>
</tbody>
</table>
To connect to this database... | Use this provider...
--- | ---
Microsoft Access | • Microsoft Office Access Database Engine OLE DB Provider (recommended)
• Microsoft Jet OLE DB Provider
If the Microsoft Office Access Database Engine OLE DB Provider is not available in the list, make sure that you have installed either Microsoft Access or the Microsoft Access Database Engine Redistributable (https://www.microsoft.com/en-us/download/details.aspx?id=54920) on your computer.

SQL Server | • Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) - this is the recommended OLE DB provider. In order for this provider to appear in the list, it must be downloaded from https://docs.microsoft.com/en-us/sql/connect/oledb/download-oledb-driver-for-sql-server?view=sql-server-ver15 and installed.
• Microsoft OLE DB Provider for SQL Server (OLEDBSQL)
• SQL Server Native Client (SQLNCLI)

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 ADO.NET, ODBC, or JDBC connection.
If the operating system has an ODBC driver to the required database, you could also use the Microsoft OLE DB Provider for ODBC Drivers, or preferably opt for an ODBC connection.

5. Having selected the provider of choice, click Next and complete the wizard.

The subsequent wizard steps are specific to the provider you chose. For SQL Server, you will need to provide or select the host name of the database server, the authentication method, the database name, as well as the database username and password. For an example, see Connecting to Microsoft SQL Server (ADO). For Microsoft Access, you will be asked to browse for or provide the path to the database file. For an example, see Connecting to Microsoft Access (ADO).

The complete list of initialization properties (connection parameters) is available in the All tab of the connection dialog box—these properties vary depending on the chosen provider and may need to be set explicitly in order for the connection to be possible. The following sections provide guidance on configuring the basic initialization properties for Microsoft Access and SQL Server databases:

• Setting up the SQL Server Data Link Properties
• Setting up the Microsoft Access Data Link Properties
14.2.3.1 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.

14.2.3.2 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 (https://www.altova.com/databasespy).

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.
4. Click **Connect**.

### 14.2.3.3 Setting up the SQL Server Data Link Properties

When you connect to a Microsoft SQL Server database through ADO, you may need to set the following connection properties in the **All** tab of the Data Link Properties dialog box.

<table>
<thead>
<tr>
<th>Property</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Integrated Security</strong></td>
<td>If you selected the <strong>SQL Server Native Client</strong> data provider on the</td>
</tr>
<tr>
<td></td>
<td><strong>Provider</strong> tab, set this property to a space character.</td>
</tr>
<tr>
<td><strong>Persist Security Info</strong></td>
<td>Set this property to <strong>True</strong>.</td>
</tr>
</tbody>
</table>
14.2.3.4 Setting up the Microsoft Access Data Link Properties

When you connect to a Microsoft Access database through ADO, you may need to set the following connection properties in the All tab of the Data Link Properties dialog box.

![Data Link Properties dialog box](image)

<table>
<thead>
<tr>
<th>Property</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data Source</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>Jet OLEDB:System Database</strong></td>
<td>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 &quot;workgroup information file&quot; error, locate the workgroup information file (System.MDW) applicable to your user profile, and set the property value to the path of the System.MDW file.</td>
</tr>
</tbody>
</table>
### 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;UserID=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.
14.2.4.1 Creating a Connection String in Visual Studio

In order to connect to a data source using ADO.NET, a valid database connection string is required. The following instructions show you how to create a connection string from Visual Studio.

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.

5. Click **Connect**.
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.
14.2.4.2 Sample ADO.NET Connection Strings

To set up an ADO.NET connection, you need to select an ADO.NET provider from the database connection dialog box and enter a connection string (see also Setting up an ADO.NET Connection). Sample ADO.NET connection strings for various databases are listed below under the .NET provider where they apply.

.NET Data Provider for Teradata
This provider can be downloaded from Teradata website (https://downloads.teradata.com/download/connectivity/net-data-provider-for-teradata). A sample connection string looks as follows:

```
Data Source=ServerAddress;User Id=user;Password=password;
```

.NET Framework Data Provider for IBM i
This provider is installed as part of IBM i Access Client Solutions - Windows Application Package. A sample connection string looks as follows:

```
DataSource=ServerAddress;UserID=user;Password=password;DataCompression=True;
```

For more information, see the ".NET Provider Technical Reference" help file included in the installation package above.

.NET Framework Data Provider for MySQL
This provider can be downloaded from MySQL website (https://dev.mysql.com/downloads/connector/net/). A sample connection string looks as follows:

```
Server=127.0.0.1;Uid=root;Pwd=12345;Database=test;
```


.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: https://www-01.ibm.com/support/docview.wss?uid=swg21429586.


Oracle Data Provider for .NET (ODP.NET)
The installation package which includes the ODP.NET provider can be downloaded from the Oracle website (see http://www.oracle.com/technetwork/topics/dotnet/downloads/index.html). A sample connection string looks as follows:

```
Data Source=DSORCL;User Id=user;Password=password;
```

Where DSORCL is the name of the data source which points to an Oracle service name defined in the tnsnames.ora file, as described in Connecting to Oracle (ODBC).

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

14.2.4.3 ADO.NET Support Notes
The following table lists known ADO.NET database drivers that are currently not supported or have limited support in MobileTogether Designer.

<table>
<thead>
<tr>
<th>Database</th>
<th>Driver</th>
<th>Support notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>All databases</td>
<td>.Net Framework Data Provider for ODBC</td>
<td>Limited support. Known issues exist with Microsoft Access connections. It is recommended to use ODBC direct connections instead.</td>
</tr>
<tr>
<td></td>
<td>.Net Framework Data Provider for OleDb</td>
<td>Limited support. Known issues exist with Microsoft Access connections. It is recommended to use ADO direct connections instead.</td>
</tr>
<tr>
<td>Firebird</td>
<td>Firebird ADO.NET Data Provider</td>
<td>Limited support. It is recommended to use ODBC or JDBC instead.</td>
</tr>
<tr>
<td>Informix</td>
<td>IBM Informix Data Provider for</td>
<td>Not supported. Use DB2 Data Server</td>
</tr>
</tbody>
</table>
Database | Driver | Support notes
--- | --- | ---
Oracle | .Net Framework Data Provider for Oracle | Limited support. Although this driver is provided with the .NET Framework, its usage is discouraged by Microsoft, because it is deprecated.
PostgreSQL | - | No ADO.NET drivers for this vendor are supported. Use a native connection instead.
Sybase | - | No ADO.NET drivers for this vendor are supported.

### 14.2.5 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 native, OLE DB, or JDBC-driven connections.

To connect to a database through ODBC, first you need to create an ODBC data source name (DSN) on the operating system. This step is not required if the DSN has already been created, perhaps by another user of the operating system. The DSN represents a uniform way to describe the database connection to any ODBC-aware client application on the operating system, including MobileTogether Designer. DSNs can be of the following types:

- System DSN
- User DSN
- File DSN

A *System* data source is accessible by all users with privileges on the operating system. A *User* data source is available to the user who created it. Finally, if you create a *File DSN*, the data source will be created as a file with the .dsn extension which you can share with other users, provided that they have installed the drivers used by the data source.

Any DSNs already available on your machine are listed by the database connection dialog box when you click **ODBC connections** on the ODBC connections dialog box.
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).
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.

14.2.5.1 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.
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).

14.2.6 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. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application Options; b) The JVM path found in the Windows registry; c) The JAVA_HOME environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC drivers from the database vendor must be installed. These may be JDBC drivers installed as part of a database client installation, or JDBC libraries (.jar files) downloaded separately, if available and supported by the database, see also Database Connection Examples.
Connecting to a Data Source

Connecting to SQL Server via JDBC with Windows credentials

If you connect to SQL Server through JDBC with Windows credentials (integrated security), note the following:

- The sqljdbc_auth.dll file included in the JDBC driver package must be copied to a directory that is on the system PATH environment variable. There are two such files, one for the x86 and one for x64 platform. Make sure that you add to the PATH the one that corresponds to your JDK platform.
- The JDBC connection string must include the property integratedSecurity=true.


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 CLASPATH 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.

<table>
<thead>
<tr>
<th>Database</th>
<th>JDBC Connection URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firebird</td>
<td>jdbc:firebirdsql://&lt;host&gt;[:&lt;port&gt;]/&lt;database path or alias&gt;</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>jdbc:db2://hostName:port/databaseName</td>
</tr>
<tr>
<td>IBM DB2 for i</td>
<td>jdbc:as400://[host]</td>
</tr>
<tr>
<td>IBM Informix</td>
<td>jdbc:informix-sqli://hostName:port/databaseName:INFORMIXSERVER=myserver</td>
</tr>
<tr>
<td>MariaDB</td>
<td>jdbc:mariadb://hostName:port/databaseName</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>jdbc:sqlserver://hostName:port;databaseName=name</td>
</tr>
<tr>
<td>MySQL</td>
<td>jdbc:mysql://hostName:port/databaseName</td>
</tr>
<tr>
<td>Oracle</td>
<td>jdbc:oracle:thin:@hostName:port:SID</td>
</tr>
<tr>
<td>Oracle XML DB</td>
<td>jdbc:oracle:oci:@hostName:port:service</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>jdbc:postgresql://hostName:port/databaseName</td>
</tr>
<tr>
<td>Progress OpenEdge</td>
<td>jdbc:datadirect:openedge://host:port;databaseName=db_name</td>
</tr>
<tr>
<td>Sybase</td>
<td>jdbc:sybase:Tds:hostName:port/databaseName</td>
</tr>
<tr>
<td>Teradata</td>
<td>jdbc:teradata://databaseServerName</td>
</tr>
</tbody>
</table>

**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.
14.2.6.1 Configuring the CLASSPATH

The CLASSPATH environment variable is used by the Java Runtime Environment (JRE) or the Java Development Kit (JDK) to locate Java classes and other resource files on your operating system. When you connect to a database through JDBC, this variable must be configured to include the path to the JDBC driver on your operating system, and, in some cases, the path to additional library files specific to the database type you are using.

The following table lists sample file paths that must be typically included in the CLASSPATH variable. Importantly, you may need to adjust this information based on the location of the JDBC driver on your system, the JDBC driver name, as well as the JRE/JDK version present on your operating system. To avoid connectivity problems, check the installation instructions and any pre-installation or post-installation configuration steps applicable to the JDBC driver installed on your operating system.

<table>
<thead>
<tr>
<th>Database</th>
<th>Sample CLASSPATH entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firebird</td>
<td>C:\Program Files\Firebird\Jaybird-2.2.8-JDK_1.8\jaybird-full-2.2.8.jar</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc.jar;C:\Program Files (x86)\IBM\SQLLIB\java\db2jcc_license_cu.jar;</td>
</tr>
<tr>
<td>IBM DB2 for i</td>
<td>C:\jt400\jt400.jar;</td>
</tr>
<tr>
<td>IBM Informix</td>
<td>C:\Informix_JDBC_Driver\lib\ifxjdbc.jar;</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>C:\Program Files\Microsoft JDBC Driver 4.0 for SQL Server\sqljdbc_4.0\enu\sqljdbc.jar</td>
</tr>
<tr>
<td>MariaDB</td>
<td>&lt;installation directory&gt;\mariadb-java-client-2.2.0.jar</td>
</tr>
<tr>
<td>MySQL</td>
<td>&lt;installation directory&gt;\mysql-connector-java-version-bin.jar;</td>
</tr>
<tr>
<td>Oracle</td>
<td>ORACLE_HOME\jdbc\lib\ojdbc6.jar;</td>
</tr>
<tr>
<td>Oracle (with XML DB)</td>
<td>ORACLE_HOME\jdbc\lib\ojdbc6.jar; ORACLE_HOME\LIB\xmlparserv2.jar; ORACLE_HOME\RDBMS\jlib\xdb.jar;</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>&lt;installation directory&gt;\postgresql.jar</td>
</tr>
<tr>
<td>Progress OpenEdge</td>
<td>%DLC%\java\openedge.jar; %DLC%\java\pool.jar;</td>
</tr>
<tr>
<td></td>
<td>Note: Assuming the Progress OpenEdge SDK is installed on the machine, %DLC% is the directory where OpenEdge is installed.</td>
</tr>
<tr>
<td>Sybase</td>
<td>C:\sybase\jConnect-7_0\classes\jconn4.jar</td>
</tr>
<tr>
<td>Teradata</td>
<td>&lt;installation directory&gt;\tdgssconfig.jar; &lt;installation directory&gt;\terajdbc4.jar</td>
</tr>
</tbody>
</table>
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 (;).

14.2.7 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 macOS server, no drivers are required to be installed on the target server as well.

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.

14.2.8 Setting up a SQLite Connection

SQLite (https://www.sqlite.org/index.html) 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.

14.2.8.1 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**.

### 14.2.8.2 Foreign Key Constraints

When you connect to an existing SQLite database from MobileTogether Designer, or when you create a new one, foreign key constraints are enabled by default. Foreign key constraints help preserve the integrity of data in your database. For example, when foreign keys are enabled, it is not possible to delete a record from a table if it has dependencies in another table.

In certain cases, you may want to temporarily override this behavior and disable foreign keys, perhaps, in order to update or insert multiple rows of data without getting data validation errors. To explicitly disable foreign keys before connecting to the SQLite database, select the **Disable Foreign Keys** option available on the database connection wizard.

!["Connect to SQLite" wizard page](image)

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 into the database, or creating “orphaned” rows. (An example of an “orphaned” row would be an address in the "addresses" table not linked to any person in the "person" table, because the person was deleted but its associated address was not.)

### 14.2.9 Using a Connection from Global Resources

Altova Global Resources are portable references to files, folders, and databases. 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.

14.2.10 Database Connection Examples

This section includes examples for connecting to a database from MobileTogether Designer through ADO, ODBC, or JDBC. The ADO.NET connection examples are listed separately, see Sample ADO.NET Connection Strings. For instructions about establishing a native connection to PostgreSQL and SQLite, see Setting up a PostgreSQL Connection and Setting up a SQLite Connection, respectively.

Note the following:
• The instructions may differ if your Windows configuration, network environment and the database client or server software are not the same as the ones described in each example.
• For most database types, it is possible to connect using more than one data access technology (ADO, ADO.NET, ODBC, JDBC) or driver. The performance of the database connection, as well as its features and limitations will depend on the selected driver, database client software (if applicable), and any additional connectivity parameters that you may have configured outside MobileTogether Designer.

14.2.10.1 Connecting to Firebird (JDBC)

This example illustrates how to connect to a Firebird database server through JDBC.

Prerequisites:

• JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application Options; b) The JVM path found in the Windows registry; c) The JAVA_HOME environment variable.
• Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
• The Firebird JDBC driver must be available on your operating system (it takes the form of a .jar file which provides connectivity to the database). The driver can be downloaded from the Firebird website (https://www.firebirdsql.org/). This example uses Jaybird 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.

14.2.10.2 Connecting to Firebird (ODBC)

This example illustrates how to connect to a Firebird 2.5.4 database running on a Linux server.

Prerequisites:

- The Firebird database server is configured to accept TCP/IP connections from clients.
- The Firebird ODBC driver must be installed on your operating system. This example uses the Firebird ODBC driver version 2.0.3.154 downloaded from the Firebird website (https://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 (https://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.

   ![Create an ODBC DSN]

   Select a Driver and click on either User or System to determine what kind of DSN you want to create.
   Firebird/InterBase(r) driver

   User DSN  System DSN  Cancel

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:

<table>
<thead>
<tr>
<th>Data Source Name (DSN)</th>
<th>Enter a descriptive name for the data source you are creating.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Enter the server host name or IP address, followed by a colon,</td>
</tr>
<tr>
<td></td>
<td>followed by the database alias (or path). In this example, the</td>
</tr>
<tr>
<td></td>
<td>host name is firebirdserv, and the database alias is products,</td>
</tr>
<tr>
<td></td>
<td>as follows:</td>
</tr>
<tr>
<td></td>
<td>firebirdserv:products</td>
</tr>
<tr>
<td></td>
<td>Using a database alias assumes that, on the server side, the</td>
</tr>
<tr>
<td></td>
<td>database administrator has configured the alias products to</td>
</tr>
<tr>
<td></td>
<td>point to the actual Firebird (.fdb) database file on the server</td>
</tr>
<tr>
<td></td>
<td>(see the Firebird documentation for more details).</td>
</tr>
<tr>
<td></td>
<td>You can also use the server IP address instead of the host</td>
</tr>
<tr>
<td></td>
<td>name, and a path instead of an alias; therefore, any of the</td>
</tr>
<tr>
<td></td>
<td>following sample connection strings are valid:</td>
</tr>
<tr>
<td></td>
<td>firebirdserver:/var/Firebird/databases/butterflies.fdb</td>
</tr>
</tbody>
</table>
|                        | 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.

<table>
<thead>
<tr>
<th>Client</th>
<th>Enter the path to the fbclient.dll file. By default, this is the bin subdirectory of the Firebird installation directory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Account</td>
<td>Enter the database user name supplied by the database administrator (in this example, PROD_ADMIN).</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the database password supplied by the database administrator.</td>
</tr>
</tbody>
</table>

6. Click OK.

14.2.10.3 Connecting to IBM DB2 (JDBC)

This example illustrates how to connect to an IBM DB2 database server through JDBC.

Prerequisites:

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application Options; b) The JVM path found in the Windows registry; c) The JAVA_HOME environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK. This example uses Oracle's OpenJDK 11.0 64-bit, and, consequently, the 64-bit version of MobileTogether Designer.
- The JDBC driver (one or several .jar files that provide connectivity to the database) must be available on your operating system. This example uses the JDBC driver available after installing the IBM Data Server Client version 10.1 (64-bit). For the JDBC drivers to be installed, choose a Typical installation, or select this option explicitly on the installation wizard.
If you did not change the default installation path, the required .jar files will be in the `C:\Program Files\IBM\SQLLIB\java` directory after installation.

- You need the following database connection details: host, port, database name, username, and password.

To connect to IBM DB2 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. This examples refers to `C:\Program Files\IBM\SQLLIB\java\db2jcc.jar`. You may need to refer to the `db2jcc4.jar` driver, depending on the database server version. For driver compatibility, refer to IBM documentation (http://www-01.ibm.com/support/docview.wss?uid=swg21363866). 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.ibm.db2.jcc.DB2Driver`. This entry becomes available only if a valid .jar file path was found either in the "Classpaths" text box, or in the operating system's CLASSPATH environment variable (see the previous step).
Enter the username and password of the database user in the corresponding text boxes.

6. Enter the JDBC connection string in the **Database URL** text box. Make sure to replace the connection details with the ones applicable to your database server.

```text
jdbc:db2://hostName:port/databaseName
```

7. Click **Connect**.

### 14.2.10.4 Connecting to IBM DB2 (ODBC)

This example illustrates how to connect 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 (JDBC)

This example illustrates how to connect to an IBM DB2 for i database server through JDBC.

Prerequisites:

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application Options; b) The JVM path found in the Windows registry; c) The JAVA_HOME environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK. This example uses Oracle’s OpenJDK 11.0 64-bit, and, consequently, the 64-bit version of MobileTogether Designer.
- The JDBC driver (one or several .jar files that provide connectivity to the database) must be available on your operating system. This example uses the open source Toolbox for Java/JTOpen version 9.8 (http://jt400.sourceforge.net/). After you download the package and unpack to a local directory, the required .jar files will be available in the lib subdirectory.
- You need the following database connection details: host, username, and password.

To connect to IBM DB2 for i 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. In this example, the required .jar file is at the following path: C:\jdbc\jtopen_9_8\lib\400.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.ibm.as400.access.AS400JDBCDriver. This entry becomes available only if a valid .jar file path was found either in the "Classpaths" text box, or in the operating system’s CLASSPATH environment variable (see the previous step).
5. Enter the username and password of the database user in the corresponding text boxes.
6. Enter the JDBC connection string in the Database URL text box. Make sure to replace host with the host name or IP address of your database server.
7. Click Connect.

14.2.10.6 Connecting to IBM DB2 for i (ODBC)

This example illustrates how to connect 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).

![Driver Selection Screen]

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**.

### 14.2.10.7 Connecting to IBM Informix (JDBC)

This example illustrates how to connect to an IBM Informix database server through JDBC.

**Prerequisites:**

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application **Options**; b) The JVM path found in the Windows registry; c) The **JAVA_HOME** environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC driver (one or several .jar files that provide connectivity to the database) must be available on your operating system. In this example, IBM Informix JDBC driver version 3.70 is used. For the driver's installation instructions, see the documentation accompanying the driver or the "IBM Informix JDBC Driver Programmer's Guide".
- You have the following database connection details: host, name of the Informix server, database, port, username, and password.
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 "Classpaths" 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.
14.2.10.8 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.

### 14.2.10.9 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 it has limitations, 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 in UNC format, for example, \myserver\mynetworkshare\Reports\Revenue.accdb, where myserver is the name of the server and mynetworkshare is the name of the network share.

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, and set the value of the Jet OLEDB: System database property to the path of the System.MDW file.

14.2.10.10 Connecting to Microsoft SQL Server (ADO)

This example illustrates how to connect to a SQL Server database through ADO. These instructions are applicable when you use the recommended Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL),

Before following these instructions, make sure that you have downloaded and installed the provider above on your workstation. The ADO provider must match the platform of MobileTogether Designer (32-bit or 64-bit).

If you would like to use other ADO providers such as SQL Server Native Client (SQLNCLI) or Microsoft OLE DB Provider for SQL Server (SQLOLEDB), the instructions are similar, but these providers are deprecated and thus not recommended. Also, for the connection to be successful with a deprecated provider, you may need to set additional connection properties as described in Setting up the SQL Server Data Link Properties.

The Microsoft OLE DB Provider for SQL Server (SQLOLEDB) is known to have issues with parameter binding of complex queries like Common Table Expressions (CTE) and nested SELECT statements.

To connect to SQL Server:

1. Start the database connection wizard.
2. Select Microsoft SQL Server (ADO), and then click Next. The list of available ADO providers is displayed. In this example, the Microsoft OLE DB Driver for SQL Server is used. If it's not in the list, make sure that it is installed on your computer, as mentioned above.

3. Click Next. The Data Link Properties dialog box appears.
4. Select or enter the name of the database server, for example, SQLSERV01. If you are connecting to a named SQL Server instance, the server name looks like SQLSERV01\SOMEINSTANCE.

5. If the database server was configured to allow connections from users authenticated on the Windows domain, select Windows Authentication. Otherwise, select SQL Server Authentication, clear the Blank password check box, and enter the database credentials in the relevant boxes.

6. Select the Allow saving password check box and the database to which you are connecting (in this example, "Nanonull").
14.2.10.11 Connecting to Microsoft SQL Server (ODBC)

This example illustrates how to connect to a SQL Server database through ODBC.

Prerequisites:

- Download and install the Microsoft ODBC Driver for SQL Server from the Microsoft website, see https://docs.microsoft.com/en-us/SQL/connect/odbc/download-odbc-driver-for-sql-server. This example uses Microsoft ODBC Driver 17 for SQL Server to connect to a SQL Server 2016 database. You might want to download a different ODBC driver version, depending on the version of SQL Server where you want to connect. For information about ODBC driver versions supported by your SQL Server database, refer to the driver's system requirements.
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 the driver from the list. Note that the driver appears in the list only after it has been installed.

5. Click User DSN (or System DSN if you are creating a System DSN).

Creating a System DSN requires that MobileTogether Designer be run as an administrator. Therefore, in order to create a System DSN, cancel the wizard, make sure that you run MobileTogether Designer as an administrator, and perform the steps above again.

6. Enter a name and, optionally, a description to identify this connection, and then select from the list the SQL Server to which you are connecting (SQLSERV01 in this example).
7. If the database server was configured to allow connections from users authenticated on the Windows domain, select **With Integrated Windows authentication**. Otherwise, select one of the other options, as applicable. This example uses **With SQL Server authentication**... , which requires that the user name and password be entered in the relevant boxes.
8. Optionally, select the **Change the default database to** check box and enter the name of the database to which you are connecting (in this example, **Sandbox**).
9. Click **Next** and, optionally, configure additional parameters for this connection.
10. Click **Finish**. A confirmation dialog box listing the connection details opens.
11. Click **OK**. The data source now appears in the list of **User** or **System** data sources, as configured, for example:

14.2.10.12 Connecting to MySQL (ODBC)

This example illustrates how to connect 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 Connector/ODBC 8.0.
Prerequisites:

- MySQL ODBC driver must be installed on your operating system. Check the MySQL documentation for the driver version recommended for your database server version (see https://dev.mysql.com/downloads/connector/odbc/).
- You have the following database connection details: host, database, port, username, and password.

If you installed MySQL Connector/ODBC for 64-bit platform, make sure to install MobileTogether Designer for 64-bit platform as well.

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).

If you installed MobileTogether Designer 64-bit, then the 64-bit ODBC drivers are shown in the list. Otherwise, the 32-bit ODBC drivers are shown. See also Viewing the Available ODBC Drivers.

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.

### 14.2.10.13 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:**

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom
Connecting to a Data Source

JVM path you may have set in application Options; b) The JVM path found in the Windows registry; c) The JAVA_HOME environment variable.

- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The Oracle Instant Client Package (Basic) must be available on your operating system. The package can be downloaded from the official Oracle website. This example uses Oracle Instant Client Package version 12.1.0.2.0, for Windows 32-bit and, consequently, Oracle JDK 32-bit.
- You have the following database connection details: host, port, service name, username, and password.

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\ojdbc7.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 "Classpaths" 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.

<table>
<thead>
<tr>
<th>Classpaths:</th>
<th>C:\jdbc\instantclient_12_1\ojdbc7.jar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver:</td>
<td>oracle.jdbc.driver.OracleDriver</td>
</tr>
<tr>
<td>Username:</td>
<td>johndoe</td>
</tr>
<tr>
<td>Password:</td>
<td>********</td>
</tr>
<tr>
<td>Database URL:</td>
<td>jdbc:oracle:thin:@//ora12c:1521/orcl12c</td>
</tr>
</tbody>
</table>

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.
14.2.10.14 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 the ODBC Data Source administrator in Windows, you can do so, and then select it when prompted by the wizard. For more information about ODBC data sources, see Setting up an ODBC Connection.

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.

7. Click Connect.
Connecting to a Data Source

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.

14.2.10.15 Connecting to PostgreSQL (ODBC)

This example illustrates how to connect 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 11.0) 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.
- You have the following database connection details: server, port, database, user name, and password.

To set up a connection to PostgreSQL using ODBC:

1. Start the database connection wizard.
2. Click ODBC Connections.
3. Select the User DSN option.
4. Click Create a new DSN and select the driver from the drop-down list. If no PostgreSQL driver is available in the list, make sure that the PostgreSQL ODBC driver is installed on your operating system, as mentioned in the prerequisites above.
5. Click User DSN.
6. Fill in the database connection credentials (these must be supplied by the database owner), and then click **Save**.

The connection is now available in the list of ODBC connections. To connect to the database, you can either double-click the connection or select it, and then click **Connect**.

14.2.10.16 Connecting to Progress OpenEdge (JDBC)

This example illustrates how to connect to a Progress OpenEdge 11.6 database server through JDBC.

**Prerequisites**

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application **Options**; b) The JVM path found in the Windows registry; c) The **JAVA_HOME** environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The operating system’s **PATH** environment variable must include the path to the **bin** directory of the JRE or JDK installation directory, for example `C:\Program Files (x86)\Java\jre1.8.0_51\bin`.
- The Progress OpenEdge JDBC driver must be available on your operating system. In this example, JDBC connectivity is provided by the **openedge.jar** and **pool.jar** driver component files available in `C:\Progress\OpenEdge\java` as part of the OpenEdge SDK installation.
- You have the following database connection details: host, port, database 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 "Classpaths" 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.
14.2.10.17 Connecting to Progress OpenEdge (ODBC)

This example illustrates how to connect 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).

![Create an ODBC DSN]

Select a Driver and click on either User or System to determine what kind of DSN you want to create.

- **Progress OpenEdge 11.6 Driver**

  - **User DSN**
  - **System DSN**
  - **Cancel**

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**.

![ODBC Progress OpenEdge Wire Protocol Driver Setup]

- **Data Source Name**: my_progress_dsn
- **Host Name**: localhost
- **Port Number**: 8910
- **Database Name**: oetpsdev
- **User ID**: altova

7. Click **OK**. The new data source now appears in the list of ODBC data sources.
14.2.10.18 Connecting to Sybase (JDBC)

This example illustrates how to connect to a Sybase database server through JDBC.

Prerequisites:

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application Options; b) The JVM path found in the Windows registry; c) The JAVA_HOME environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- Sybase jConnect component must be installed on your operating system (in this example, jConnect 7.0 is used, installed as part of the Sybase Adaptive Server Enterprise PC Client installation). 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.

8. Click Connect.
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 "Classpaths" 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.
14.2.10.19 Connecting to Teradata (JDBC)

This example illustrates how to connect to a Teradata database server through JDBC.

Prerequisites:

- JRE (Java Runtime Environment) or Java Development Kit (JDK) must be installed. This may be either Oracle JDK or an open source build such as Oracle OpenJDK. MobileTogether Designer will determine the path to the Java Virtual Machine (JVM) from the following locations, in this order: a) The custom JVM path you may have set in application Options; b) The JVM path found in the Windows registry; c) The JAVA_HOME environment variable.
- Make sure that the platform of MobileTogether Designer (32-bit, 64-bit) matches that of the JRE/JDK.
- The JDBC driver (one or more .jar files that provide connectivity to the database) must be available on your operating system. In this example, Teradata JDBC Driver 16.20.00.02 is used. For more information, see https://downloads.teradata.com/download/connectivity/jdbc-driver.
- You have the following database connection details: host, 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**.
14.2.10.20 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 https://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.
14.3 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 source

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 \(\textit{(screenshot above)}\), click \textit{Add/Edit SELECT Statement}. This displays the Enter a SQL SELECT Statement dialog \(\textit{(screenshot below)}\).

2. Enter the \textit{SELECT} statement you want and give the statement a name. In the screenshot below, we select, with the \textit{WHERE} clause, rows that have \textit{id}=20. We give the statement a name of \textit{Vienna Offices}.

3. Click \textit{Add SELECT Statement} to add the statement. The \textit{SELECT} statement appears in the Source pane \(\textit{(see screenshot below)}\).

4. Click \textit{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.
14.4 Editing DB Data

This section:

- About OriginalRowSet
- Editing DB data in tables and other controls
- Updating, inserting, appending, and deleting nodes
- The DB Execute action and $MT_DBExecute_Result variable
- Primary Keys in MobileTogether Designer

About OriginalRowSet

In order for data to be edited and saved, the tree of the page source must also include an OriginalRowSet element, which is a copy of the RowSet element. The original data is saved in the OriginalRowSet element, while edited data is saved in the RowSet element. When the page source is saved, the difference between the two trees, OriginalRowSet and RowSet, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to OriginalRowSet so that OriginalRowSet contains the newly saved DB data, and the modification process can be repeated.

Note the following points:

- The OriginalRowSet element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command Create OriginalRowSet.
- The Create OriginalRowSet command is enabled for database type ($DB) root nodes. It is a toggle command that creates/removes an OriginalRowSet data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the OriginalRowSet structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function mt-db-original-row.

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 page source node that is to be edited. Do this by dragging the page source node onto the control.
- If you use a table with repeating rows, use the option to automatically include Append/Delete controls when the table is created (see screenshot below).
The advantage of using a table with repeating rows that are linked to the repeating Row elements of a DB page source is that when a table row is added, a DB row is also automatically added. For more information, see the description of how to work with tables. 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:

```xquery
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
```
14.5 Saving Data to the DB

This section:

- Saving on the basis of solution progress
- The Save action
- The DB Execute action
- Filtering the columns to save
- About OriginalRowSet
- Committing Transactions

Saving on the basis of solution progress
The context menu of a $DB root node has a Save Data command (screenshot below) which enables the data source represented by the root node to be updated at different points during the progress of the solution. The options are described below. If the default option, Not Automatically, is selected, then data is saved only when the Save action of an event is triggered.

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

- **On Every Page Leave**: Data in the tree is saved every time a page containing that tree is exited.
- **On Any Solution Finish**: Data in the tree is saved when the solution is exited, no matter at what point or how the solution is exited.
- **On Last Submit**: Data in the tree is saved when the workflow progresses as designed, from first page to last page, and when the last Submit button is tapped. If this option is selected and the solution is exited before the last Submit button is tapped, then data in the tree will not be saved.
- **Not Automatically**: The tree will not automatically be saved. You must use the Save, 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 OnSubmitButtonClick 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 page source. You can specify which columns can be updated or can take inserted values. (Updates refer to modified data in existing row elements; inserted values refer to data in newly added row elements.) By default, the Insert and Update options of each column are selected together as a pair. If, however, you wish to specify different options for a column’s Insert and Update options, check the Use separate filter settings for insert and update statements check box. Attributes with empty values can be converted to NULL values in the DB by checking that column’s NULL check box. Note that missing attributes will always be saved as NULL.

Columns that cannot be updated (because they are user-defined, fixed-value, or calculated-value) will not have an Insert, Update, or NULL option check box. In the screenshot above, the ID column cannot be updated because it holds fixed values. Deselect the columns you do not want to update.

You can specify the order in which deletions, updates, and insertions occur by selecting the desired order in the combo box at the bottom of the dialog.

If you wish to reset the Save settings so that all columns are updated, click Reset to default.

About OriginalRowSet
In order for data to be edited and saved, the tree of the page source must also include an OriginalRowSet element, which is a copy of the RowSet element. The original data is saved in the OriginalRowSet element, while edited data is saved in the RowSet element. When the page source is saved, the difference between the two trees, OriginalRowSet and RowSet, is calculated, and the page source is updated on the basis of the difference. If the modification is successful, then the modified data is copied to OriginalRowSet so that OriginalRowSet contains the newly saved DB data, and the modification process can be repeated.
Note the following points:

- The `OriginalRowSet` element is not created by default in the tree of the DB page source. To create it, right-click the root node of the DB page source and toggle on the command `Create OriginalRowSet`.
- The `Create OriginalRowSet` command is enabled for database type ($DB) root nodes. It is a toggle command that creates/removes an `OriginalRowSet` data structure that contains the original data of the page source.
- Till the time modified data is saved to the DB, the original DB data is retained in the `OriginalRowSet` structure. This ensures that the original DB data is still available in the tree.
- To retrieve the original data of a DB row that has been modified but not yet saved, use the XPath function `mt-db-original-row`.

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.
14.6 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.

![DB Execute action screenshot](image)

In this section, we describe how to insert, update, delete, and save data using DB Execute. The command to modify DB data is specified in the SQL statement of the action (see screenshot above). For a detailed description of the settings of the DB Execute action, see the section **Actions > Database > DB Execute**. 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:

```sql
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.
14.7 Displaying DB Data

Displaying DB data in tables and other controls

DB data can be displayed in a control by creating a page source link from the control to a page source node. Typically the best way to display DB data is in a table with repeating rows. Drag a table control into the design and create a new table as a repeating table (see screenshot below). Then, drag controls into the cells of the table and make page source links to the column nodes of the DB row. For an actual demonstration of how this works, see the tutorial, Database-And-Charts.

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](#).
14.8 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. 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.

<table>
<thead>
<tr>
<th>Database</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firebird 2.5.4, 3.0</td>
<td></td>
</tr>
<tr>
<td>IBM DB2 8.x, 9.1, 9.5, 9.7, 10.1, 10.5, 11.5</td>
<td></td>
</tr>
<tr>
<td>IBM Db2 for i 6.1, 7.1, 7.2, 7.3, 7.4</td>
<td>Logical files are supported and shown as views.</td>
</tr>
<tr>
<td>IBM Informix 11.70, 12.10, 14.10</td>
<td></td>
</tr>
<tr>
<td>MariaDB 10.2, 10.3, 10.4, 10.5</td>
<td></td>
</tr>
<tr>
<td>Microsoft Access 2003, 2007, 2010, 2013, 2016, 2019</td>
<td>At the time of writing (early September 2019), there is no Microsoft Access Runtime available for Access 2019. You can connect to an Access 2019 database from Altova products only if Microsoft Access 2016 Runtime is installed and only if the database does not use the “Large Number” data type.</td>
</tr>
<tr>
<td>Microsoft Azure SQL Database</td>
<td>SQL Server 2016 codebase</td>
</tr>
<tr>
<td>MySQL 5.0, 5.1, 5.5, 5.6, 5.7, 8.0</td>
<td></td>
</tr>
<tr>
<td>Oracle 9i, 10g, 11g, 12c, 18, 19</td>
<td></td>
</tr>
<tr>
<td>PostgreSQL 8.0, 8.1, 8.2, 8.3, 9.0, 9.1, 9.2.1, 9.4, 9.5, 9.6, 10, 11, 12.1, 13</td>
<td>PostgreSQL connections are supported both as native connections and driver-based connections through interfaces (drivers) such as ODBC or JDBC. Native connections do not</td>
</tr>
<tr>
<td>Database</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>require any drivers.</td>
</tr>
<tr>
<td>Progress OpenEdge 11.6</td>
<td></td>
</tr>
<tr>
<td>SQLite 3.33.0</td>
<td>SQLite connections are supported as native, direct connections to the SQLite database file. No separate drivers are required.</td>
</tr>
<tr>
<td>Sybase ASE 15, 16</td>
<td></td>
</tr>
<tr>
<td>Teradata 16</td>
<td></td>
</tr>
</tbody>
</table>

### 14.8.1 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.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Command</th>
<th>Does this</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>master</td>
<td></td>
</tr>
</tbody>
</table>
### 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.

## 14.8.2 Connecting to Data Sources

In order to query a database, you have to first connect to the required database, and then select the required data source and root object from among multiple existing connections.

---

### Connecting to a database

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.
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.

14.8.3 Browser Pane

The Browser pane provides an overview of objects in the selected database. This overview includes database constraint information, such as whether a column is a primary or foreign key. In IBM DB2 version 9 and higher databases, the Browser additionally shows registered XML schemas in a separate folder (see screenshot below).

This section describes the following:

- The layouts 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.

### 14.8.4 Query Pane: Description

The Query pane is an intelligent SQL editor for entering queries to the selected database. After entering the query, clicking the **Execute command** 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:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Command</th>
<th>Does this</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Execute Query" /></td>
<td>Execute Query</td>
<td>Executes currently selected SQL statement. If script contains multiple statements and none is selected, then all are executed.</td>
</tr>
<tr>
<td><img src="image" alt="Import SQL File" /></td>
<td>Import SQL File</td>
<td>Opens an SQL file in the SQL Editor.</td>
</tr>
<tr>
<td><img src="image" alt="Export SQL File" /></td>
<td>Export SQL File</td>
<td>Saves SQL queries to an SQL file.</td>
</tr>
<tr>
<td><img src="image" alt="Undo" /></td>
<td>Undo</td>
<td>Undoes an unlimited number of edits in SQL Editor.</td>
</tr>
<tr>
<td><img src="image" alt="Redo" /></td>
<td>Redo</td>
<td>Redoes an unlimited number of edits in SQL Editor.</td>
</tr>
<tr>
<td><img src="image" alt="Options" /></td>
<td>Options</td>
<td>Open the Options dialog of SQL Editor.</td>
</tr>
<tr>
<td><img src="image" alt="Open Query in DatabaseSpy" /></td>
<td>Open Query in DatabaseSpy</td>
<td>Opens the SQL script in Altova's DatabaseSpy application.</td>
</tr>
</tbody>
</table>

---

**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.
14.8.5 Query Pane: Working With

After connecting to a database, an SQL script can be entered in the SQL Editor and executed. This section describes:

- How an SQL script is entered in the SQL Editor.
- How the script is executed in the Database Query window.

The following icons are referred to in this section:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Execute Query</th>
<th>Executes currently selected SQL statement. If script contains multiple statements and none is selected, then all are executed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>🎥</td>
<td>Import SQL File</td>
<td>Opens an SQL file in the SQL Editor.</td>
</tr>
</tbody>
</table>

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.

14.8.6 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).

<table>
<thead>
<tr>
<th>CID</th>
<th>INFO</th>
<th>HISTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>&lt;xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot; ?&gt;</td>
<td>[NULL]</td>
</tr>
<tr>
<td>1001</td>
<td>&lt;xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot; ?&gt;</td>
<td>[NULL]</td>
</tr>
<tr>
<td>1002</td>
<td>&lt;xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot; ?&gt;</td>
<td>[NULL]</td>
</tr>
<tr>
<td>1003</td>
<td>&lt;xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot; ?&gt;</td>
<td>[NULL]</td>
</tr>
<tr>
<td>1004</td>
<td>&lt;xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot; ?&gt;</td>
<td>[NULL]</td>
</tr>
<tr>
<td>1005</td>
<td>&lt;xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot; ?&gt;</td>
<td>[NULL]</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Go to Statement</th>
<th>Highlights the statement in the SQL Editor that produced the current result.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Find</strong></td>
<td>Finds text in the Results pane. XML document content is also searched.</td>
</tr>
</tbody>
</table>

**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.
15 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.
15.1 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).
Defining Global Resources

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.

15.1.1 Files

The Global Resource dialog for Files (screenshot below) is accessed via the Add | File 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 | File 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.
Defining the alias
Define the alias (its name and configurations) as follows:

1. **Give the alias a name:** Enter the alias name in the Resource Alias text box.
2. **Add configurations:** The Configurations pane will have, by default, a configuration named Default (see screenshot above), which cannot be deleted or renamed. You can add as many additional configurations as you like by: (i) clicking the Add Configuration or Add Configuration as Copy icons, and (ii) giving the configuration a name in the dialog that pops up. Each added configuration will be shown in the Configurations list. In the screenshot above, two additional configurations, named Long and Short, have been added to the Configurations list. The Add Configuration as Copy command enables you to copy the selected configuration and then modify it.
3. **Select a resource type for each configuration:** Only a File resource may be selected as a global resource for a MobileTogether design.
4. **Select a file for the resource type:** If the resource is a directly selected file, browse for the file in the Resource File Selection text box.
5. **Define multiple configurations if required:** You can add more configurations and specify a resource for each. Do this by repeating Steps 3 and 4 above for each configuration. You can add a new configuration to the alias definition at any time.
6. **Save the alias definition:** Click OK to save the alias and all its configurations as a global resource. The global resource will be listed under Files in the Manage Global Resources dialog.
15.1.2 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.

### 15.1.3 Databases

In the Global Resource dialog for Databases (screenshot below), you can add a database resource as follows:

<table>
<thead>
<tr>
<th>Alias name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource alias: MyDatabase</td>
</tr>
</tbody>
</table>

#### Databases Configuration

**Settings for configuration "AlternativeDB"**

- **Database**: C:\Workarea\Examples\altova.mdb
  - **Database Kind**: MS Access
  - **Import Kind**: ADO

**MapForce-specific execution parameters**

- **DataSource**: C:\Workarea\Examples\altova.mdb
- **Catalog**: altova
- **Provider**: Microsoft.Jet.OLEDB.4.0
- **JDBCDatabaseURL**: jdbc:odbc://DRIVER=Microsoft Access
- **JDBCDriver**: sun.jdbc.odbc.JdbcOdbcDriver

[Image of the Global Resource dialog for Databases]
Global Resource dialog icons

- **Add Configuration:** Pops up the Add Configuration dialog in which you enter the name of the configuration to be added.
- **Add Configuration as Copy:** Pops up the Add Configuration dialog in which you can enter the name of the configuration to be created as a copy of the selected configuration.
- **Delete:** Deletes the selected configuration.

Saving unverified connection info

If the connection information you enter cannot correctly access the database you want, then this might be because the connection information is correct only when the solution is deployed to the server but does not work from the local machine. If the connection does not work from the local machine (on which you are defining the global resource), then the Connection Info dialog (screenshot below) appears.

![Connection Info Dialog](screenshot)

Database connection failed. No additional properties retrieved. Please verify and correct connection info.

**Database vendor:** Unknown

**Connection string:** DSN=ORACLE12_mobileto;

Note: if you are sure that connection info is correct but database connection cannot be established, you can use "Save info as is" to store connection info without further checks.

- **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.
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.

```
Choose Root Object

You can set the root object for current database data source configuration. Once you have set the root object for a configuration you cannot change it. If you do not set it you will have to define it in the application that you are using and will be permitted to change it.

master

Set Root Object  Skip
```

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.
15.2 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.

15.2.1 Assigning Files and Folders

In certain usage scenarios, file-type and folder-type global resources can be used to specify the file or folder to use. For example, in the Page Sources Pane, 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. 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).

15.2.2 Assigning Databases

When a command is executed that imports data or a data structure from a DB, you can select the option to use a global resource (screenshot below).
In the Connection dialog (screenshot above), all the database-type global resources that have been defined in the currently active Global Resources XML File 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.

### 15.2.3 Changing the Active Configuration

One configuration of a global resource can be active at any time. This configuration is called the active configuration, and it is active application-wide. This means that the active configuration is active for all global resources aliases in all currently open files and data source connections. If an alias does not have a configuration with the name of the active configuration, then the default configuration of that alias will be used.

As an example of how to change configurations, consider the case in which a file has been assigned via a global resource with multiple configurations. Each configuration maps to a different file. So, which file is selected depends on which configuration is selected as the application's active configuration.

Switching the active configuration can be done in the following ways:
Via the menu command **Tools | Active Configuration**. Select the configuration from the command's submenu.

In the combo box of the Global Resources toolbar (*screenshot below*), select the required configuration.

In this way, by changing the active configuration, you can change source files that are assigned via a global resource.
16 Simulation

You can simulate the workflow on the currently selected preview device. To preview (simulate) a solution on a different device, change the preview device.

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 device**

You can select the device that you want to simulate in the Preview Device combo box. To see a simulation on a different platform or different device, choose the new device and re-start the simulation.

**Simulation language**

The simulation language for designer 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.
16.1 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 toolbar
The toolbar of the Simulation pane contains the following buttons (from left to right):

- **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**.
- **Change Orientation**: You can switch the view between landscape and portrait.
- **Simulator Options**: Drops down a menu of simulator options (described below under Simulator options).
- **Simulate Return to App (Reopen)**: Enabled when page refreshes have been defined to occur on page reopens. Refreshes the page during simulations.
- **Stop at Next Error**: Stops the simulation at the next XPath error and displays the XPath expression in the XPath Debugger.
- **Stop at Next Breakpoint**: Stops the simulation at the next breakpoint (which may be on an action or XPath expression) and opens the appropriate Debugger.
- **Stop at Next Action**: Stops the simulation at the action(s) of the next event that is triggered and displays the action(s) in the Actions Debugger.
- **Stop Recording Test Case**: When the recording of a test case is started, the simulator is opened and the recording runs in the simulator. Clicking this button: (i) stops the recording of the test case, and (ii) opens the Recorded Test Case Confirmation dialog, in which you specify the name of the recorded test case.
- **Playback Next Step**: Plays back the next step in the currently running test-case playback. This button is enabled when playback is set to step-by-step. See the Playback Options section of Playing Back a Test Case.
- **Record Snapshot**: Records a snapshot of the test case that is being recorded. This button is enabled when the recording of snapshots has been set to manual (not automatic). See the Recording Options section of Recording a Test Case.
- **Push Notifications**: Opens the Manage Recorded Push Notification (Simulation) dialog. See Simulating Push Notifications for a description of usage.
Simulation

- **Purchase:** Provides a list of all the in-app purchase products that are available for purchase, and enables you to quickly purchase one or more of these. The list of products is obtained from the [In-app Purchase XML data file for simulations](#). See [In-App Purchases](#) for more information.

- **Purchased:** Provides a list of all the in-app purchase products that are listed as purchased (that is, as a Purchase element) in the [In-app Purchase XML data file for simulations](#). If you select one of these purchased products, then this product together with all its purchase properties is considered, in the simulation, to have been purchased. If you were to subsequently run the [Query Purchases](#) action or [Restore Purchases](#) action, then the products that you selected will be returned (by the action) as having been purchased. See [In-App Purchases](#) for more information.

- **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 the **Page Refresh option** has been set to Manual, then this button becomes visible. On clicking **Refresh Page**, the page is updated with the changed data.

**Simulator options**

On clicking the **Simulator Options** toolbar button, a dropdown list of options for the simulator is displayed (screenshot below). Each option is described below. Default settings are specified in the [Simulation 1 tab](#) of the Options dialog. Note that, if you change a setting in the simulator, the new setting becomes the default and will be shown as such in the [Simulation 1 tab](#).

- **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 actions, and will allow you to more easily analyze other messages and aspects of the workflow.

- **Prevent Server Access:** When selected, disables access to the server, and so allows you to test the
solution’s behavior in a server-connection-error scenario. When unselected, server access is allowed. For more information about this feature, see Server Connection Errors.

- **Is Server Purchased:** For simulations in the designer and for trial runs on client, simulates that MobileTogether Server licenses have been purchased. For simulations on the server, the actual purchase-state of licenses on the server is returned.

- **Simulate WIFI:** Sets the `mt-connected-via-wifi` 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:** When selected, enables NFC functionality so that NFC actions can be executed. Actual NFC data is supplied to the simulator via NFC sample files.

- **Simulate GPS:** When selected, enables geolocation functionality so that geolocation features can be tested. 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.

- **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 2 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.

- **Simulate DB Read Structure:** When toggled on, the simulator takes the DB structure from the XML file that is specified in the Simulation 2 tab of the Options dialog. For relevant information, see the DB Read Structure action.

- **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.

- **Set Default Options:** Resets Simulation pane options to their default settings.

---

**Page Sources pane: options and features**

The following options are available in the Page Sources toolbar.

- **Evaluate XPath:** Opens the XPath/XQuery Window, in which you can evaluate XPath expressions. XPath expressions can also be evaluated from the Styles & Properties Pane while the simulation is running.
- **Clear Persistent Data and Restart**: Clears persistent data and restarts the simulation.
- **Restart Simulation**: Restarts the simulation at any time.
- **Search**: Enables searches for text in the page sources in the Page Sources pane, starting from the root element of page sources.
- In the Page Sources pane you can use copy-paste to copy parts of the tree to other locations in the tree. This is useful if you wish to copy data, such as DB records, in order to add more data for the simulation. The copied nodes are available only for the duration of the simulation.
- If you right-click a node of a page source and select **Load XML**, then the entire page source is replaced by the XML file you select. Note that, if the structure of the loaded XML file does not match the structure expected of this page source, then those parts of the design that are based on this page source will contain errors.
- If you right-click a node of a page source and select **Save XML**, then the entire page source is saved to the selected XML file.
- If you right-click a node of a page source, you can copy the XPath locator expression of that node to the clipboard by selecting the **Copy XPath** context menu command.

**Editing the XML trees of page sources in the Simulator**

The XML trees in the simulator display the XML data of the various page sources and how these values change as the simulation progresses. You can edit the XML trees directly in the Simulator using cut/copy/paste/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](https://www.altova.com/xmlSpy).
- **Overwrite $XML structure based on this tree**: Overwrites the structure of a page source with the structure of the XML tree in the simulator.
16.2 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.
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 toolbar
The toolbar of the Simulation pane contains the following buttons (from left to right):

• 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.
• Change Orientation: You can switch the view between landscape and portrait.
• Simulator Options: Drops down a menu of simulator options (described below under Simulator options).
• Simulate Return to App (Reopen): Enabled when page refreshes have been defined to occur on page reopens. Refreshes the page during simulations.
• Stop at Next Error: Stops the simulation at the next XPath error and displays the XPath expression in the XPath Debugger.
• Stop at Next Breakpoint: Stops the simulation at the next breakpoint (which may be on an action or XPath expression) and opens the appropriate Debugger.
• Stop at Next Action: Stops the simulation at the action/s of the next event that is triggered and displays the action/s in the Actions Debugger.
• Stop Recording Test Case: When the recording of a test case is started, the simulator is opened and the recording runs in the simulator. Clicking this button: (i) stops the recording of the test case, and (ii) opens the Recorded Test Case Confirmation dialog, in which you specify the name of the recorded test case.
• Playback Next Step: Plays back the next step in the currently running test-case playback. This button is enabled when playback is set to step-by-step. See the Playback Options section of Playing Back a Test Case for a description of usage.
• Record Snapshot: Records a snapshot of the test case that is being recorded. This button is enabled when the recording of snapshots has been set to manual (not automatic). See the Recording Options section of Recording a Test Case for more information.
• Push Notifications: Opens the Manage Recorded Push Notification (Simulation) dialog. See Simulating Push Notifications for a description of usage.
• Purchase: Provides a list of all the in-app purchase products that are available for purchase, and enables you to quickly purchase one or more of these. The list of products is obtained from the In-app Purchase XML data file for simulations. See In-App Purchases for more information.
• Purchased: Provides a list of all the in-app purchase products that are listed as purchased (that is, as a Purchase element) in the In-app Purchase XML data file for simulations. If you select one of these purchased products, then this product together with all its purchase properties is considered, in the simulation, to have been purchased. If you were to subsequently run the Query Purchases action or Restore Purchases action, then the products that you selected will be returned (by the action) as having been purchased. See In-App Purchases for more information.
• 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 OnSubmitButtonClick.
• Refresh Page: If the Page Refresh option has been set to Manual, then this button becomes visible.
On clicking **Refresh Page**, the page is updated with the changed data.

**Simulator options**

On clicking the **Simulator Options** toolbar button, a dropdown list of options for the simulator is displayed (screenshot below). Each option is described below. Default settings are specified in the **Simulation 1 tab** of the Options dialog. Note that, if you change a setting in the simulator, the new setting becomes the default and will be shown as such in the **Simulation 1 tab**.

- **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 actions, and will allow you to more easily analyze other messages and aspects of the workflow.
- **Prevent Server Access**: When selected, disables access to the server, and so allows you to test the solution's behavior in a server-connection-error scenario. When unselected, server access is allowed. For more information about this feature, see **Server Connection Errors**.
- **Is Server Purchased**: For simulations in the designer and for trial runs on client, simulates that MobileTogether Server licenses have been purchased. For simulations on the server, the actual purchase-state of licenses on the server is returned.
- **Simulate WiFi**: Sets the `mt-connected-via-wifi` 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**: When selected, enables NFC functionality so that NFC actions can be executed. Actual NFC data is supplied to the simulator via NFC sample files.
- **Simulate GPS**: When selected, enables geolocation functionality so that geolocation features can be tested. 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 2 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 2 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.
- **Simulate DB Read Structure**: When toggled on, the simulator takes the DB structure from the XML file that is specified in the Simulation 2 tab of the Options dialog. For relevant information, see the DB Read Structure action.
- **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.
- **Set Default Options**: Resets Simulation pane options to their default settings.

**Page Sources pane: options and features**

The following options are available in the Page Sources toolbar.

- **Evaluate XPath**: Opens the XPath/XQuery Window, in which you can evaluate XPath expressions. XPath expressions can also be evaluated from the Styles & Properties Pane while the simulation is running.
- **Clear Persistent Data and Restart**: Clears persistent data and restarts the simulation.
- **Restart Simulation**: Restarts the simulation at any time.
- **Search**: Enables searches for text in the page sources in the Page Sources pane, starting from the root element of page sources.
- **In the Page Sources pane you can use copy-paste to copy parts of the tree to other locations in the tree. This is useful if you wish to copy data, such as DB records, in order to add more data for the simulation. The copied nodes are available only for the duration of the simulation.
- **If you right-click a node of a page source and select Load XML, then the entire page source is replaced by the XML file you select. Note that, if the structure of the loaded XML file does not match the structure expected of this page source, then those parts of the design that are based on this page source will contain errors.
- **If you right-click a node of a page source and select Save XML, then the entire page source is saved to the selected XML file.**
• If you right-click a node of a page source, you can copy the XPath locator expression of that node to the clipboard by selecting the **Copy XPath** context menu command.

**Note:** If you have problems connecting to the server, try checking the settings on the server. See the [MobileTogether Server user manual](#) for information about how to do this.

**Editing the XML trees of page sources in the Simulator**

The XML trees in the simulator display the XML data of the various page sources and how these values change as the simulation progresses. You can edit the XML trees directly in the Simulator using cut/copy/past/delete and drag-and-drop. (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 structure of the XML tree in the simulator.

**Server IP address and network firewall settings**

Your server can have a public IP address (accessible over the Internet) and/or a private IP address (accessible within a private network; for example, via WiFi within a company network). If a mobile client device tries to connect via the Internet using the server's private IP address, then the connection will not work. This is because the private IP address is not known on the Internet and cannot be resolved. If a client device uses a private IP address, then the client device must already have access to the private network.

To ensure that the server can be accessed, do one of the following:

- Provide the server with a public IP address so that it can be reached via the Internet. On the client device, use this public IP address to access the server.
- If you use a firewall and install MobileTogether Server on a server with a private IP address (inside the private network), then use the network firewall to forward requests sent to a public IP-address/port-combination to your MobileTogether Server server. On the client device, use the public IP address.

You must also ensure that the firewall is configured to allow access to the server port used for MobileTogether Client communication. The ports used by MobileTogether Server are specified in the Settings page of the Web UI of MobileTogether Server ([see the MobileTogether Server user manual](#)). On the client device, this is the port that must be specified as the server port to access.

**Tip:** Port 80 is usually open on most firewalls by default. So, if you are having difficulties with firewall settings and if port 80 is not already bound to some other service, you could specify port 80 as the MobileTogether Server port for client communication.
16.3 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 of MobileTogether Designer. Simulation on the client assumes that your mobile device can connect to your PC over wireless LAN. To run a client simulation, carry out the steps listed below. (For client simulations of compiled apps, see the box below.)

1. **Client: Set up a new server**
   1. Start MobileTogether on your client device and click the MobileTogether app's **Settings | Server** icon.
   2. Add a new server and enter the following data: a connection name, the IP address of the machine on which MobileTogether Designer is running, and the port number on which MobileTogether Designer is listening (by default this is 8083). (In MobileTogether Designer, click **Project | Trial Run on Client** to check at which port MobileTogether Designer will be listening.)
   3. Save the server settings.

2. **PC: Start the simulation**
   1. If MobileTogether Server is running on your PC and uses the same port as the designer's listening port, then stop MobileTogether Server running as a service.
   2. In MobileTogether Designer, you can change MobileTogether Designer's listening port. Do this in the **Trial Run on Client** tab of the Options dialog (**Tools | Options**).
   3. In MobileTogether Designer, select **Project | Trial Run on Client**. This does two things: (i) opens, in designer, a Trial Run on Client dialog that will show the page sources of the solution, and (ii) opens the designer's connection to the client. Note that communication between designer and client starts only when Trial Run on Client has been started in MobileTogether Designer.
   4. On the client device, refresh the view to see a listing of the designs (the design list) that are currently open in MobileTogether Designer.

3. **Client: Run the simulation**
   1. On the client, select from the design list the solution that you want to test.
   2. This opens a prompt in MobileTogether Designer asking if you want to start the solution. Click **Yes**. The trial run of the solution starts on the client and the solution's page sources now appear in the dialog box of MobileTogether Designer.
   3. Click **Back/Return** to stop the current solution. A prompt appears asking if you want to stop. Click **Yes**.

**Note:** A design file can be used by only one client at a time for client simulation.

---

**Client simulations of compiled apps**

The procedure for simulating a trial run of a compiled app is similar to that for solutions described above. The main difference is that the designer's connection details will have already been specified in the compiled app (see **Screen 3 of the Code-Generation wizard**). As a result, these details do not need to be set on the client.
Additionally, after you have compiled an app-store app, the design can be changed as often as you like. You will not need to recompile the app to run a client simulation. The app will connect to MobileTogether Designer and use the currently open version of the design.

The steps to set up client simulations of a compiled app are as follows:

1. In MobileTogether Designer, go to the Trial Run on Client tab of the Options dialog and make sure that the listening port specified there is the same as that you specified in the compiled app (in Screen 3 of the Code-Generation wizard).
2. In MobileTogether Designer, open the project file (.mtd) of the compiled app that you want to test, and edit it as required.
3. Start the compiled app on your device. The compiled app will use the project file in MobileTogether Designer (including the last edit even if this was not saved), and the pages sources and simulation messages will be displayed in MobileTogether Designer.

Simulation language

The simulation language for designer 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.

XML trees of page sources in the Simulator

The XML trees in the simulator display the XML data of the various page sources and how these values change as the simulation progresses. The context menu of XML trees in the simulator offers the following features:

- **Copy XPath**: Copies to the clipboard an XPath locator expression that locates the selected tree node.
- **Save XML**: Saves the structure and data of an XML tree to any location you like.
- **View in XMLSpy**: Opens the XML tree in Altova's XMLSpy program.
- **Overwrite $XML structure based on this tree**: Overwrites the structure of a page source with the structure of the XML tree in the simulator.
- **Send XML to Designer**: Sends all client XML data to the designer, enabling you to analyze the solution's XML data.
16.4 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 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 page source tree nodes (including the $GEOLOCATION 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 **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.

  ```xml
  <Root>
  ```
<Geolocations>
  <Geolocation name="Buckingham Palace">
    <Location>
      Latitude="51.501364"
      Longitude="-0.14189"
      Provider="gps"
    </Location>
    <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/>
</Geolocations>
</Root>

- All attributes of the Geolocation XML file

<Root>
  <Geolocations>
    <Geolocation name="">
      <Location>
        AccuracyHorizontal=""
        AccuracyVertical=""
        Altitude=""
        Latitude=""
        Longitude=""
        MagneticHeading=""
        Provider=""
        Speed=""
        Time=""
      </Location>
      ...
    </Geolocation>
  </Geolocations>
</Root>
Geolocation input string formats:

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from +90 to -90 (N to S). Longitude values range from +180 to -180 (E to W).

Note: If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow ("=") while unit indicators that are escaped are highlighted in blue ("=").

- Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)
  \( D^\circ M'.S.SS''N/S \quad D^\circ M'.S.SS''W/E \)
  Example: \( 33^\circ 55.11''N \quad 22^\circ 44.55.25''W \)

- Degrees, minutes, decimal seconds, with prefixed sign (+/-); the plus sign for (N/E) is optional
  \( +/-D^\circ M'.S.SS'' \quad +/-D^\circ M'.S.SS'' \)
  Example: \( 33^\circ 55.11'' \quad -22^\circ 44.55.25'' \)

- Degrees, decimal minutes, with suffixed orientation (N/S, E/W)
  \( D^\circ M.M'M'M'N/S \quad D^\circ M.M'M'M'W/E \)
  Example: \( 33^\circ 55.55'N \quad 22^\circ 44.44'W \)

- Degrees, decimal minutes, with prefixed sign (+/-); the plus sign for (N/E) is optional
  \( +/-D^\circ M.M'M'M' \quad +/-D^\circ M.M'M'M' \)
  Example: \( +33^\circ 55.55' \quad -22^\circ 44.44' \)

- Decimal degrees, with suffixed orientation (N/S, E/W)
**Geolocation Settings**

- **Example:** 33.33N 22.22W

  - Decimal degrees, with prefixed sign (+/-); the plus sign for (N/S E/W) is optional
    
    **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
16.5 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**

```xml
<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 />
        </NdefMessage>
        <NdefRecord />
        ...
        <NdefRecord />
      </Root>
    </NFC>
    <NFC/>
    ...
    <NFC/>
  </NFCS>
  </Root>
</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.
16.6 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.

![Record PN Data Dialog](image)

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 from 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.
16.7 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
<?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>
```
16.8 Calendar Sample Files

In order to simulate the calendar of a device you can use a sample calendar file that has the XML structure shown in the listing below. You can then specify, in the Simulation tab of the Options dialog, 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.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Root>
  <Calendar Id="1" Name="Business">
    <Event Id="1" Title="Quarterly Meeting" Start="2018-04-04" End="2018-04-04" AllDay="true()" Location="Meeting Room 2">
      <Attendee Name="Bob" Status="Accepted" Type="Required" Relationship="Speaker"/>
    </Event>
    <Event Id="2" Title="New Customer Lunch" Start="2018-05-14T12:30:00" End="2018-05-14T14:00:00" Location="Sushi Restaurant">
      <Attendee Name="Alice" Status="Accepted" Type="Optional" Relationship="Attendee"/>
    </Event>
  </Calendar>
  <Calendar Id="2" Name="Private">
    <Event Id="1" Title="Family Dinner" Start="2018-05-18T19:00:00" End="2018-05-18T23:00:00" Location="Home"/>
    <Event Id="2" Title="Summer Vacation" Start="2018-07-09" End="2018-07-22" AllDay="true()" Location="Home"/>
  </Calendar>
</Root>
```

**Note:** In the Microsoft Outlook calendar, attendee status is visible only to the organizer of the event. So you will not be able to see the attendee status of events that were not organized by you.
16.9 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`.
16.10 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 custom 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.
17 Offline Usage

Your MobileTogether solution, whether accessed using the MobileTogether Client app on a client device or an app store app, can be designed to enable a user to work offline (that is, with no active network or Internet connection). Data can be synchronized with one or more databases on the MobileTogether Server when the user reconnects to the server.

This topic: (i) lists the main settings that enable offline work; and (ii) describes how each of these settings affect offline work and can be used in a design.

The sub-topics of this section contain descriptions of different ways in which users can work offline and synchronize subsequently, when they come online.

Steps to enable offline work

To enable the user to work offline, use one or more of the following settings according to what your design needs are.

- In the Project Settings, set Server Access to On demand. This ensures that the server will be contacted (for downloading or uploading data), only when the solution initiates such an action.
- In the context menu of page source trees, set both Load Data and Save Data to Not Automatically.
- Use the $PERSISTENT page source to store data on the client.
- If you want to download data to the client, use a Reload action.
- Use a Save action or DB Execute action to upload data to the server.

The steps above are general indicators. For a better understanding, see the discussion of these settings below.

For a description of some scenarios of offline use, see the examples in the sub-topics of this section.

Settings that enable offline work

The following settings determine when and how data is transferred between client and server. These settings effectively configure offline usage and data synchronization. Set their values according to the needs of your solution.

Project settings

The Server Access setting is a key setting for determining offline use. It takes one of three values: (i) Always, (ii) On Demand, (iii) Never. The main difference between Always and On Demand is that in the case of Always data transfer between server and client occurs continually, whereas in the case of On Demand data transfers occur only when explicitly specified via an action. One consequence of using On Demand is that data trees are not automatically updated at solution start. This difference can be seen in the example solutions 02-DisplayRecords.mtd and 03-DisplayOnDemand.mtd, where Server Access has been set, respectively, to Always and On Demand.

Load data

The Load Data setting is available on the root node of page sources. It is not available on $PERSISTENT page sources. Load Data takes one of three values: (i) On first use, (ii) On every page, (iii) Not automatically. The value of the setting can be assigned when the page source is created; it can be changed subsequently via the context menu of the source's root node. If you want to work offline and to ensure that the client does not automatically connect to the server (on solution start or when a new page loads), then set
Load Data to Not Automatically. Given this setting, the server will not be contacted for automatic data loading, and the client can stay offline. To load data from the server, you would have to implement an explicit action to load data, for example, a Load File action, Reload action, or a DB Execute action.

Save data
The Save Data setting is available on the root node of any page source that is linked to a data file or a DB. It is not available on $PERSISTENT page sources. Save Data takes one of four values: (i) On every page leave, (ii) On any solution finish, (iii) On last submit, (iv) Not automatically. The value of the setting can be assigned when the page source is created, and it can be changed subsequently via the context menu of the source's root node. If you want to work offline and ensure that the client does not automatically connect to the server (on a page leave, a solution finish, or a last submit), then set Save Data to Not Automatically. In this case, the server will not be contacted for automatic data saving, and the client can stay offline. To save data to the server, you would have to implement an action such as the Save action.

By setting Server Access to On Demand and Load Data and Save Data to Not Automatically, you effectively set the client device to be offline.
17.1 Enter Data Offline and Upload

The example solution 01-AddRecord.mtd (start screen shown below) is located in the following (My) Documents folder:
Altova\MobileTogetherDesigner7\MobileTogetherDesignerExamples\Tutorials\OfflineUsage. Open the file in MobileTogether Designer and run a simulation (F5) to see how it works.

The solution enables records to be entered on a client device, one record at a time, with each being saved to a SQLite database on the server, Addresses.sqlite, before the next record is entered. During editing, the record is stored in the $PERSISTENT page source on the client (see screenshot below). The solution's Server Access is restricted by the On Demand setting, as a result of which the solution will only communicate with the server when communication is required and is explicitly specified in the solution's workflow. Till such time, the solution is offline and the record's data is entered while the solution is offline. When the user clicks Upload to database now, the solution on the client connects with the server to upload the record to the SQLite database on the server.

Key settings
The key settings of the 01-AddRecord.mtd solution and for working offline are discussed below.

The Load Data setting
The Load Data setting is not applicable for this solution because the solution has only one page source, $PERSISTENT, which is on the client and for which, therefore, no data loading is required. (Since we want to display only the new record in the client—and not all the DB records—no other page source is needed to hold the records of the DB.)

The Save Data setting
The Save Data setting is not applicable to $PERSISTENT page sources.
The Server Access setting
The Server Access setting is set to On Demand. As a result the server is contacted only when the new record has to be uploaded to the server.

This happens when Upload to database now is clicked. This button's OnButtonClicked event triggers the following actions:

1. A DB Execute action that executes an SQL statement to insert the data of the $PERSISTENT tree as a new record on the server DB.
2. An Update Node action to reset the nodes of the $PERSISTENT tree, each to the empty string. This is required to enable data entry of the next record.

Note: In the design, click the Event Actions icon (circled in red in the screenshot above) to see how the actions for updating the database (described above) have been defined.
17.2 Connect to Server on Demand

The example solutions `02-DisplayRecords.mtd (start screen shown below left)` and `03-DisplayOnDemand.mtd (start screen shown below right)` are located in the following `(My) Documents` folder: `Altova\MobileTogetherDesigner7\MobileTogetherDesignerExamples\Tutorials\OfflineUsage`. Open the files in MobileTogether Designer and run simulations `F5` to see how they work.

Both solutions differ from the previous solution, `01-AddRecord.mtd`, in that they only display the data of the server-based SQLite database `Addresses.sqlite`; there is no mechanism to add records to the DB. The solutions have been deliberately simplified to focus on the display of data. The difference between the two is in their start screens: While that of `02-DisplayRecords.mtd (2)` contains data downloaded from the server, that of `03-DisplayOnDemand.mtd (3)` does not display any server data. In `03-DisplayOnDemand.mtd (3)`, the solution is offline; it only goes online to download server data when the `Refresh` button (circled red in screenshot below right) is clicked.

Notice in the screenshots of the designs that the cells of the table in `02-DisplayRecords.mtd (2)` are linked to the `$DB1` page source (screenshot below left), whereas the cells of the table in `03-DisplayOnDemand.mtd (3)`, are linked to the `SPERSISTENT` page source (screenshot below right). These page sources—to `$DB1` (the page source linked to the database on the server) and `SPERSISTENT` (the page source on the client)—cause data from the respective page sources to be displayed in the table.
In (2), data is downloaded from the server to $DB1 on solution start and displayed right away in the start screen.

In (3), data is downloaded from the server to $DB1 only when the Refresh button is clicked. It is only then that the data is copied to $PERSISTENT and, because of the page source link in the table, displayed in the table. This is why the table is empty till the Refresh button is clicked.

Key settings
The key settings of both solutions are compared in the following table.

<table>
<thead>
<tr>
<th>02-DisplayRecords.mtd (2)</th>
<th>03-DisplayOnDemand.mtd (3)</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Load Data</strong> $\rightarrow$ On First Use</td>
<td><strong>Load Data</strong> $\rightarrow$ Not Automatically</td>
<td>In (2), DB data is displayed on solution start. In (3), it isn’t.</td>
</tr>
<tr>
<td><strong>Server Access</strong> $\rightarrow$ Always</td>
<td><strong>Server Access</strong> $\rightarrow$ On Demand</td>
<td>In (3), ensures that the server is contacted only when the Refresh button is clicked.</td>
</tr>
<tr>
<td>No action for OnPageRefresh</td>
<td><strong>OnPageRefresh</strong> $\rightarrow$ loads DB data from server</td>
<td>In (3), DB data is loaded from server on page refresh. The Refresh button is enabled only when an action is defined for the event.</td>
</tr>
</tbody>
</table>
The Load Data setting
The reason that (2) contains downloaded data while (3) does not is due to the **Load Data** setting, which in (2) is *On First Use* and in (3) is *Not Automatically*. In (2) data is loaded when the solution starts.

The Server Access setting
The **Server Access** setting (**Always**, **On Demand**, or **Never**) does not affect the data download on solution start (which depends on the **Load Data** setting). But the **Server Access** setting lets you decide when the server will be contacted. The **On Demand** setting provides you with greater control of the design and the communication process. For example, in (3), the table data is downloaded from the server only when the solution's **Refresh** button is clicked. In the design, this is specified by defining the following set of actions for the **OnPageRefresh** event:

1. **Reload** the $DB1 page source from the server. The server is contacted and the data is downloaded.
2. **Delete the Row nodes** of the $PERSISTENT tree. This is preparatory to adding to $PERSISTENT the just updated row nodes of $DB1 (see previous step). Note that the data is removed only from the $PERSISTENT page source on the client; the database on the server is not affected.
3. **Update the Persistent tree’s Root node** with the just updated row nodes of the $DB1 tree. This is required because it is the nodes of the $PERSISTENT page source (and not those of $DB1) that are displayed in the table (enclosed in green in screenshot below).

<table>
<thead>
<tr>
<th>Display Server Records on Demand</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
</tr>
<tr>
<td>Row ($PERSISTENT)</td>
</tr>
<tr>
<td>First ($PERSISTENT)</td>
</tr>
</tbody>
</table>

The point to note is that in (3) the server is accessed and data downloaded only when the **Reload** action is triggered.
17.3 Edit Data Offline and Synchronize

The example solutions [04-EditRecords.mtd](#) and [05-EditRecordsOnStart.mtd](#) are examples of how to edit data offline and go online solely to save data to a database on the server.

They are located in the following (My Documents) folder: Altova\MobileTogetherDesigner7\MobileTogetherDesignerExamples\Tutorials\OfflineUsage. Open the files in MobileTogether Designer and run simulations (F5) to see how they work.

Both solutions save data to a SQLite database on the server, AddressesIndexed.sqlite. The solution [04-EditRecords.mtd](#) is different from [05-EditRecordsOnStart.mtd](#) in one respect only: that records from the SQLite database are not displayed in the start screen of the former solution, but are displayed in the start screen of the latter.

Description of the example solutions

The example solution [04-EditRecords.mtd](#) (design shown in screenshot below) displays the DB data in a table and allows users to edit the data. Fields of the table are linked to the \$PERSISTENT page source. This means (i) that it is the contents of \$PERSISTENT that are displayed in the table, and (ii) that edits you make in the table are stored in \$PERSISTENT.

When the solution [04-EditRecords.mtd](#) starts, the table is empty (see screenshot below). This is because the table has \$PERSISTENT as its page source and \$PERSISTENT is empty. Note also that the \$DB1 page source is empty. This is because it has been set to Load Data = Not Automatically. In the Page Sources pane, right-click \$DB1 to see the value of its Load Data setting.
In the solution, the following data edits are possible:

- A new record (row) can be added to the table and the fields of the record can be edited. To add a new row, click the Plus icon located to the right of the last row (see screenshots above and below). After you add a new row, the data of the row is stored in $PERSISTENT (see screenshot below).

- A new record (row) can be added to the table and the fields of the record can be edited. To add a new row, click the Plus icon located to the right of the last row (see screenshots above and below). After you add a new row, the data of the row is stored in $PERSISTENT (see screenshot below).

- In the solution, a row can be deleted by clicking the its Minus icon (see screenshot above).
- The data in $PERSISTENT can be saved to the server by clicking Save to database. For an explanation of how this works, see the section Saving edited data by using primary keys and OriginalRowSets below.
- On clicking Refresh (circled red in screenshot below), all data rows are downloaded from the server to $DB1 and copied from there to $PERSISTENT (see the actions of the OnPageRefresh event as defined in this event's tab). Any data that was present in $PERSISTENT before the refresh will be deleted. Since the rows in $PERSISTENT are displayed in the table, the table now displays all the server records. Note that the Server Access setting for $DB1 has been set to On Demand. This ensures that the server will be contacted only when a request is made to the server, such as that for reloading database data. In effect, this means that users can work offline till they need to upload data to the server.
Key settings
The key settings of the (04-EditRecords.mtd) solution are given in the following table.

<table>
<thead>
<tr>
<th>04-EditRecords.mtd</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Data (of $DB1) = Not Automatically</td>
<td>Server DB data is not displayed on solution start.</td>
</tr>
<tr>
<td>Server Access (of $DB1) = On Demand</td>
<td>The client connects the server only when required by an action.</td>
</tr>
<tr>
<td>OnPageRefresh reloads DB data from server,</td>
<td>On page refresh; $DB1 is reloaded with data from the server, and this</td>
</tr>
<tr>
<td>deletes nodes of $PERSISTENT, and appends</td>
<td>data is copied to $PERSISTENT. As a result, the table will contain the</td>
</tr>
<tr>
<td>the new nodes of $DB1 to $PERSISTENT.</td>
<td>latest data on the server.</td>
</tr>
</tbody>
</table>

At solution start, load all data for editing
When the solution 04-EditRecords.mtd is started, the table is empty because $PERSISTENT is empty. If we want to show all the server DB data in the table, we could use the following strategy, which has been used in 05-EditRecordsOnStart.mtd. To see how this has been set up, open 05-EditRecordsOnStart.mtd and see the definition of the OnPageLoad event of the solution’s Start page.

1. Since the actions we want to carry out must occur when the solution starts, we create them on the OnPageLoad event of the solution’s Start page.
2. Reload $DB1 via the Reload action. This downloads all DB data from the server to $DB1 (when the solution starts).
3. Delete all the records of $PERSISTENT via the Delete Node(s) action.
4. Copy all the record nodes from $DB1 to $PERSISTENT, by using the Update Node(s) action. Once the data has been appended to the $PERSISTENT tree, it will automatically be displayed in the table.
This is because the controls in the table's cells have page source links to the nodes of the $PERSISTENT page source.

**Saving edited data by using primary keys and OriginalRowSets**

In both `04-EditRecords.mtd` and `05-EditRecordsOnStart.mtd`, the setting to create an OriginalRowSet node has been set to true. This setting is switched on via a toggle command in the context menu of the page source. In both our examples, whenever the page is refreshed (via the Refresh button), the $DB1 page source is reloaded from the server and two important steps are executed: (i) an OriginalRowSet node is created in $DB1 that contains an exact deep copy of the tree's RowSet node (that is, of all the rows of the server DB), and (ii) the RowSet and OriginalRowSet nodes are copied from $DB1 to the $PERSISTENT tree. This way we know what rows were originally present in the table before editing started: edited rows are in RowSet, while original rows are in OriginalRowSet.

Now if the table is edited, that is, if new rows are added or deleted, then, in $PERSISTENT, the number of rows in RowSet and OriginalRowSet will be different. Each row is identified by a unique primary key field named ID, which may not be NULL and is automatically set to an auto-incremented integer when the row is added.

When the data is saved to the server DB, the saving mechanism is as follows:

1. $DB1 is reloaded by using the Reload action.
2. From $DB1/RowSet, those rows are deleted that have the same ID as that of rows in $PERSISTENT/OriginalRowSet. This ensures that all the original rows (before editing) are deleted from $DB1/RowSet.
3. The rows of $PERSISTENT/Rowset are copied to $DB1/RowSet, with the Append Node(s) action. These are the new, the edited, and the unedited rows in $PERSISTENT/Rowset that we want to save to the server. Rows that have been deleted from the table will not be in this rowset and so will not be copied to $DB1. As a result, $DB1 will now contain only the rows that we want to save back to the server.
4. A Save action saves the data from $DB1 to the SQLite database on the server, with only modifications being saved.
18 MT Debugger

MT Debugger comprises two debuggers:

- **Actions Debugger**, which is opened when an action that has been selected for debugging is encountered during a simulation.
- **XPath Debugger**, which is built into the XPath/XQuery Window. You can access the XPath Debugger when editing XPath expressions in the design or when running simulations.

Usage overview

MT Debugger is started from within a simulation. The broad usage steps are listed below:

1. Before running the simulation, set up debugging breakpoints on actions and/or XPath expressions if you want to debug these during the simulation.
2. Start the simulation.
3. Select the **Debugger mode** you want to use: Breakpoints or Actions. In Breakpoints Mode, the simulation stops at all breakpoints (actions and XPath expressions) and opens the respective debugger. In Actions Mode, the simulation stops at the set of actions defined on the next event that is triggered in the Simulator. If no mode is selected, then the simulation proceeds without any debugging.
18.1 Debugger Modes

MT Debugger modes are set in the Simulator. There are three debugger modes:

- **Breakpoints Mode:** In this mode, the simulation will stop at breakpoints. If the breakpoint has been defined on an action, Actions Debugger will open and the action will be displayed in it. If the breakpoint is defined on an XPath expression, XPath Debugger will open and the expression will be displayed in it. You can then start debugging the action or expression.

- **Actions Mode:** In this mode, the simulation will stop at the set of actions defined on the next event that is triggered in the Simulator. The Actions Debugger will open, and the event's actions will be displayed in it.

- **Errors Mode:** In this mode, the simulation will stop when it encounters XPath errors.

Note the following points:

- Only one mode can be selected at a time.
- You can change modes during a simulation.
- After each simulation, the mode selection is removed. So the mode must be selected afresh for each simulation.

Select the debugger mode

You can select a debugger mode either before or after starting a simulation:

- Before starting the simulation: Select the mode via the the MobileTogether Designer toolbar or the Debug menu.
- After starting the simulation: Select the mode in the Simulator toolbar.

In both toolbars, the icons and commands are the same (see below).

<table>
<thead>
<tr>
<th>MT Debugger modes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop at Next Error</td>
<td>Stops the simulation at XPath errors (Errors Mode)</td>
</tr>
<tr>
<td>Stop at Next Breakpoint</td>
<td>Stops the simulation at breakpoints (Breakpoints Mode)</td>
</tr>
<tr>
<td>Stop at Next Action</td>
<td>Stops the simulation at actions of the next event that is triggered (Actions Mode)</td>
</tr>
</tbody>
</table>
18.2 Actions Debugger

The Actions Debugger enables you to debug the actions of a control event or page event. It is opened when an action that has been selected for debugging is encountered during a simulation. The debugging of actions enables you to do the following: (i) view the callstack of actions, (ii) view how the values of variables are updated during the execution of actions, and (iii) set watch expressions to test or investigate aspects of the action execution.

Usage

Usage of the Actions Debugger consists of two steps:

1. Select the action/s to be debugged
2. Start a simulation and run the Actions Debugger on the selected actions

Select the action/s to debug

Selection is done in two ways:

- By setting a breakpoint on each action you want to debug. If you then run a simulation in Breakpoints Mode, the Actions Debugger will automatically open to debug actions that have breakpoints. (In Breakpoints Mode, XPath Debugger will be opened to debug any XPath expression that contains a breakpoint.)
- By directly selecting, during a simulation, the next action to debug. This is Actions Mode.

Setting breakpoints on actions

You can set a breakpoint on an action in two ways:

- On an action’s definition, by clicking the toolbar command Toggle Breakpoint (screenshot below) to toggle the breakpoint on. To remove the breakpoint, click Toggle Breakpoint again.
During an action debugging session you might find that you want to set or remove breakpoints for future debugging sessions. This is possible without you having to close your current debugging session. In the Actions Debugger (screenshot below), select the action for which you want to set/remove a breakpoint, and click the toolbar command **Toggle Breakpoint**.

The following options are available when you place the mouse cursor over a breakpoint's symbol (the red circle):

- Click the gear symbol in the popup to make the breakpoint conditional. The condition is specified as follows: (i) via an XPath expression, and (ii) via the hit count (the number of times the action has been executed). For example, you can specify that the breakpoint is active when the hit count is equal to 3, which will result in the breakpoint being active the third time the action is executed.
- Click the outlined-circle toggle in the popup to disable/enable the breakpoint.

A plus symbol appears inside the red circle symbol of the breakpoint when the breakpoint has been made conditional. This symbol disappears when the conditionality is removed.

**Note:** You can remove a breakpoint by clicking it, or by selecting the respective action and clicking the **Toggle Breakpoint** icon in the toolbar of the pane.

### Select an action during simulation
Besides using breakpoints to select an action for debugging, you can also, during a simulation, use the simulator's **Stop at Next Action** toolbar icon (see screenshot below) to select actions for debugging. The selected actions in this case will be the actions of the next event to be triggered in the simulation.

When the next event is triggered in the simulator, the Actions Debugger will open, and you can start debugging the actions that were defined for that event. How the Actions Debugger works is explained below.
Running the Actions Debugger

The Actions Debugger can run in Breakpoints Mode (debugs actions having breakpoints) or Actions Mode (debugs actions of the next event to be triggered in the simulation).

After you have set any breakpoints (see section above), in the simulator select the debugger mode you want (Breakpoints or Actions) and then start or resume the simulation. When the simulation reaches an action selected for debugging, the Actions Debugger appears.

The Actions Debugger (screenshot below) consists of two panes:

- An upper pane, which displays the action/s that have been selected for debugging. The screenshot below, for example, shows the two actions that have been set on the OnLabelClicked event of a Label control named Label9. These actions are an Update Node action and a Go to Subpage action.
- A lower pane, which has three tabs: (i) Callstack, (ii) Watches, (iii) Variables (described below).

You can start debugging by using the toolbar icons

- The action being currently debugged is indicated by a green highlight and green arrow at left (see screenshots in this topic)
- The toolbar icons (see screenshot above) are, from left to right:
  - Resume Debugging / Go: Starts/Continues the debugging.
  - Stop Debugging: Stops debugging and closes the Actions Debugger. You will be prompted to choose whether you want to (i) stop the debugging and the simulation; (ii) stop the debugging and edit the action; (iii) cancel the stop action and continue with debugging.
  - Step Into Action (F11): Proceeds through the action execution, one step at a time.
- **Step Into XPath (Ctrl+Shift+F11):** Opens the XPath Debugger and displays the XPath expression of the action.
- **Step Out (Shift+F11):** Steps out of the current execution step, and goes to the parent step.
- **Step Over (Ctrl+F11):** Steps over descendant steps.
- **Run to Cursor:** Stops at the action currently selected by the cursor. (The currently selected action is highlighted in light blue; see screenshot above.) If a breakpoint occurs earlier, processing stops at the breakpoint/s. If the current selection cannot be reached, then processing proceeds to the end stopping only at breakpoints.
- **Toggle Breakpoint (F9):** Toggles on/off a breakpoint on the selected action. If MT Debugger is run in Breakpoints Mode, then the simulation will be stopped at the selected action to debug.

The Callstack, Watches, and Variables panes

The three tabs of the lower pane display the results of the debugging. They are described below. At each step of debugging, the results in each tab are updated. You can switch between tabs to check the various results.

The three screenshots below show the Actions Debugger during a simulation of the SubPages-And-Visibility tutorial. The debugger was in Actions Mode and the first customer, New Fashion, was clicked. This opened the Actions Debugger with the action of the OnLabelClicked Control displayed (the label containing the New Fashion text). The action was an action group named Show Orders. On clicking Step Into in the Actions Debugger, we stepped into the Show Orders action group (the actions of which are shown in the screenshots below), and stepped past the first Update Node action. The screenshots were taken on stepping into the second Update Node action. You can open the SubPages-And-Visibility tutorial, and try this out for yourself.

**Callstack pane**

Displays the actions that are executed and the results of the execution (see screenshot below). The callstack displays the current action, in which the node $XML1/Root/CustomerName has been updated with the text content of the Name element.
**Watches pane**

In the Watches pane, you can enter XPath expressions to display data, generate new data (such as, in the example below, the number of customers), and test whether certain conditions are true. Click **Add Watch** to add a new watch expression, enter the XPath expression, and press **Return**.
Variables pane

The Variables pane display all the design variables that are currently in scope, including Static Global Variables and Dynamic Local Variables. This information can be very useful for building and debugging expressions defined inside actions.
Closing the Actions Debugger

You can close the Actions Debugger in the following ways:

- Click Stop Debugging.
• Click through the action/s using the **Step Into, Step Out Of, and Step Over** toolbar buttons till the action/s of the current event have all been executed in the debugger.
• In the Evaluator, toggle off the **Actions Mode**.
18.3 XPath Debugger

You can access XPath Debugger in the following contexts:

1. When editing an XPath expression in the design. For example, when entering expressions to set values of styles and properties, you can debug expressions by testing them against an XML file that you load into the Debugger.

2. When running a simulation: Enter a new expression at any time during the simulation to debug it against an XML file that you load into the Debugger. To start the XPath Debugger, click the Evaluate XPath button in the Page Sources pane (see screenshot below).

3. When running a simulation: In the Actions Debugger, click the Step Into XPath button to debug the current action's XPath expression against the action's page sources (see screenshot below).

How to use XPath Debugger is explained in the XPath/XQuery Window section.

Setting breakpoints on XPath expressions

You can set breakpoints on XPath expressions in XPath Debugger after opening XPath Debugger via any of the three contexts listed above. Furthermore, if breakpoints have been set on expressions, you can debug these expressions directly from a simulation by starting MT Debugger in Breakpoints Mode. In this mode the simulation will stop at all breakpoints, whether they have been set on actions or XPath expressions, opening the appropriate debugger: Actions Debugger or XPath Debugger.
19 Embedded Webpage Solutions

An embedded webpage solution is a solution that is embedded in a webpage. In effect, the HTML code of the webpage will contain an IFrame element, into which the solution is loaded. Data can be exchanged between the webpage and its embedded solution with the use of JavaScript. The solution itself interacts with MobileTogether Server in the usual way, and receives data from MobileTogether Server that can then be communicated back to the webpage via JavaScript mechanisms.

The figure below left shows how the embedded solution interacts with its embedding webpage and MobileTogether Server. The screenshot below right shows a webpage that contains an embedded solution (framed in green).
Browser

```html
<html>
...<iframe>
    Solution is loaded here
</iframe>
...<html>
```

MobileTogether Server
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.
19.1  **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)*

```html
```

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.

```html
<!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)

```html
<!DOCTYPE html>
<html>
<head>
  <title>Webpage containing an embedded solution</title>
  <style>
    .resize {
      position: relative;
      padding-bottom: 56.25%; /* proportion value to aspect ratio 16:9 (9/16 = 0.5625 or 56.25%) */
      padding-top: 30px;
      height: auto; /* alternatively, try a value of 0 */
      overflow: hidden;
    }

    .resize iframe {
      position: absolute;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
    }
  </style>
</head>
<body>
  <h1>Webpage containing an embedded solution</h1>
  <p>The embedded solution is loaded into an IFrame, which is located immediately below this paragraph. CSS style properties are used to resize the IFrame according to window dimensions so that the IFrame's aspect ratio is maintained at 16:9.</p>
  <div class="resize">
    <iframe src="http://localhost:8083/run?d=/public/About" frameborder="0"></iframe>
  </div>
</body>
</html>
```
Webpage containing an embedded solution

The embedded solution is loaded into an IFrame, which is located immediately below this paragraph. CSS style properties are used to resize the IFrame according to window dimensions.

About MobileTogether

About Altova® MobileTogether®

MobileTogether is a powerful new platform to build mobile apps in record time.
19.2 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).

### 19.2.1 Posting: From Webpage to Server

A message that is sent from a webpage (via an IFrame embedded in that webpage) to the server is called an **embedded message**. It is sent in the following stages:

1. JavaScript is used to send the embedded message from the webpage to the IFrame. The embedded message is sent as a JSON object that is the first parameter of the `postMessage()` method.
2. When the embedded message reaches the IFrame, the solution serializes the message to a JSON string and sends this string to the solution's workflow on MobileTogether Server. This step is carried out automatically; you do not need to specify any processing.
3. In the workflow, the embedded message can be picked up with the `OnEmbeddedMessage` 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](https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage). 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:

```javascript
function sendMyMessage() {
    document.querySelector('iframe').contentWindow.postMessage(myJSONData, '*');
}
```

In this example, the JSON data structure will be received by the IFrame. The solution in the IFrame will send the data to the workflow as a JSON string. In the workflow, the data can be picked up by the `OnEmbeddedMessage` page event and stored in the $MT_EMBEDDEDMESSAGE JSON page source.

See [this Mozilla Developer Network documentation](https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage) for detailed information about `postMessage()`. Also see the JSON and XML examples.
19.2.2 Listening: From Server to Webpage

A message is sent from MobileTogether Server to a webpage in the following stages:

1. The workflow's Embedded Message Back 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:
   ```javascript
   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:
   ```javascript
   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).

![Screenshot of a JSON page source structure](image)

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 (the `json` node and its contents) that is sent as the message event.

```
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`. 
19.3 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).

19.3.1 Anonymous Login

Anonymous login enables the solution to be displayed in the IFrame without the user having to login or be authenticated in any way. Anonymous login is the simplest authentication method, but should only be set if the solution does not (i) allow the modification of important databases, or (ii) display information that is confidential.

To set anonymous access, do the following:

- Set the server to allow anonymous login (see the server settings for mobile client ports)
- Permit anonymous use of the solution (by setting the permissions of the workflow's container to a minimum of container=read and workflow=read, use).
19.3.2 User Login

This type of authentication requires the user to log in to MobileTogether Server with the correct credentials before the solution loads. When the call to the solution is triggered, the MobileTogether Server login page is displayed. For the log in to be authenticated, two criteria need to be fulfilled:

- The user must be a registered user of MobileTogether Server
- The user must have the appropriate permissions to access the targeted solution

For more information about registering users with MobileTogether Server and setting up their permissions, see the MobileTogether Server documentation.

19.3.3 JWT Authentication

An embedded webpage solution can also be authenticated with a JSON Web Tokens (JWT, recommended pronunciation: "jot"). Essentially, JWT authentication is processed outside the MobileTogether Server authentication system. An authenticated user is issued with a JWT, which is passed via the webpage to MobileTogether Server. On the server, the JWT is verified; it is also parsed to find out the user. If the JWT is valid, then communication between the embedded solution and the server proceeds for the user specified in the JWT.

The illustration below shows how JWT authentication works.
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):
  \[
  \text{HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), secret|privateKey)}
  \]
  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.

**19.3.3.1 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:

- **Issuer** checks whether the time of server access lies inside the validity period of the JWT. So set the issuance and expiration times as appropriate.
- **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).
- **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.

![Signed JSON Web Token](image)

**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]
19.3.3.2 Asymmetric Keys: the Public Key

If you are using asymmetric encryption for your JWT, then the encryption (JWT signing) is done with the private key, and verification is done with the public key. In order for MobileTogether Server to be able to verify the JWT, you must do the following:

In the Settings tab of MobileTogether Server, enable JWT authentication (see screenshot below), and then enter settings for:

- **Secret**: Enter the public key of a private–public pair. (If you are using symmetric encryption, enter the shared secret.)
- **Audience**: Enter the same string as that you entered for this claim when creating the JWT.

![Screenshot of JWT authentication settings](image-url)
19.4 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

19.4.1 Embedding a Solution

The HTML code listed below shows how an IFrame element is used to embed a solution. You can try out this feature by copying the HTML code below and saving it to file, and then opening the file in a browser.

```html
<!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

19.4.2 Sending/Receiving JSON Data

This section explains the working of an embedded webpage solution that uses JSON source data. A list of books in JSON format is sent from the webpage (screenshot below) to an embedded IFrame (framed in blue). Here, the list can be edited using a MobileTogether solution. On saving the changes in the IFrame, the edited book list is sent to the webpage.
Example showing how to interact with a single JSON source

The books we want to edit:

```
{
  "books": [
  {
    "author": "Mary Shelley",
    "title": "Frankenstein; or, The Modern Prometheus",
    "year": 1818,
    "pages": 280
  },
  {
    "author": "Bram Stoker",
    "title": "Dracula",
    "year": 1897,
    "pages": 302
  }
  ]
}
```

Click LOAD to load the list into the IFrame. In the IFrame, edit the list. You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that changes are propagated to the list in the webpage.

<table>
<thead>
<tr>
<th>Author</th>
<th>Mary Shelley</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Frankenstein; or, The Modern Prometheus</td>
</tr>
<tr>
<td>Year</td>
<td>1818</td>
</tr>
<tr>
<td>Pages</td>
<td>280</td>
</tr>
<tr>
<td>Author</td>
<td>Bram Stoker</td>
</tr>
<tr>
<td>Title</td>
<td>Dracula</td>
</tr>
<tr>
<td>Year</td>
<td>1897</td>
</tr>
<tr>
<td>Pages</td>
<td>302</td>
</tr>
</tbody>
</table>

Save
The embedded webpage solution consists of the HTML webpage (\texttt{jsonBooks.html}) and a MobileTogether design (\texttt{jsonBooks.mtd}). Both files are located in your (\texttt{My Documents}) MobileTogether folder: 

\texttt{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 \textbf{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.

\textbf{HTML code listing}

The HTML code listing of the file \texttt{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.

\textbf{Webpage code listing}

\begin{verbatim}
<!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 = [
      {
        "author": "Mary Shelley",
        "title": "Frankenstein; or, The Modern Prometheus",
        "year": 1818,
        "pages": 280
      },
      {
        "author": "Bram Stoker",
        "title": "Dracula",
        "year": 1897,
        "pages": 302
      }
    ];
  </script>
</head>
<body>
  <!-- This HTML code is not shown for brevity. -->
</body>
</html>
\end{verbatim}
function sendbooks() {
  document.querySelector('iframe').contentWindow.postMessage(books, '*');
}

function showbooks() {
  document.querySelector('code').innerText = JSON.stringify(books, null, ' ');
}

function receivebooks(m) {
  books = m.data.json;
  showbooks();
}

window.addEventListener('message', receivebooks);

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>
<button onclick="sendbooks()">Load</button>
</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`.

```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
    }
  ]
};
```

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`):

```javascript
document.addEventListener('DOMContentLoaded', showbooks);

function showbooks() {
  document.querySelector('code').innerText = JSON.stringify(books, null, ' ');
}
```

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):
On clicking the **Load** button:

```html
<button onclick="sendbooks()">Load</button>
```

A JavaScript function uses `postMessage()` to send the contents of the `books` variable to the IFrame.

```javascript
function sendbooks() {
    document.querySelector('iframe').contentWindow.postMessage(books, '*');
}
```

The `{books}` data 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:

```javascript
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).

```javascript
function receivebooks(m) {
    books = m.data.json;
    showbooks();
}
```

The showbooks function displays the updated book list in the webpage:

```javascript
function showbooks() {
    document.querySelector('code').innerText = JSON.stringify(books, null, ' ');
}
```

### 19.4.3 Sending/Receiving XML Data

This section explains the working of an embedded webpage solution that uses XML source data (a book list). The book list is sent from the webpage (screenshot below) to an embedded IFrame (framed in blue). Here, the list can be edited using a MobileTogether solution. On saving the changes in the IFrame, the edited book list is sent to the webpage.
Example showing how to interact with an XML source

The books we want to edit:

```xml
<books>
  <item>
    <author>Mary Shelley</author>
    <title>Frankenstein; or, The Modern Prometheus</title>
    <year>1818</year>
    <pages>280</pages>
  </item>
  <item>
    <author>Bram Stoker</author>
    <title>Dracula</title>
    <year>1897</year>
    <pages>301</pages>
  </item>
</books>
```

Click LOAD to load the book list into the iFrame. In the iFrame, edit the list. You can add, delete, and/or modify entries. Click SAVE to save changes. Notice that changes are propagated to the list in the webpage.
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

```html
<!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>
</script>
```

```xml
<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>
</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:

```javascript
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`):

```javascript
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.

```javascript
function showbooks() {
    books = new DOMParser().parseFromString(books, 'text/xml');
    document.querySelector('code').innerText = books.childNodes[0].outerHTML;
}

<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):

```html
```
**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.)

```javascript
function sendbooks() {
    // Access the contents of the iFrame's document.
    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.

The Load from String action (*screenshot above*) creates the content of `$MT_EMBEDDEDMESSAGE/json/books` as the XML page source `$books`. The structures of both these page sources have been created beforehand (*see screenshot below*).
The XML page source has been created so that the XML data can be linked to design components, thereby enabling the XML data to be edited. The design contains a repeating table of item nodes of the $books page source (see screenshot below). Cells of the table are linked, respectively, to the author, title, year, and pages page source nodes.

Because the page source has been loaded, the repeating table is populated with the data in the workflow's $MT_EMBEDDEDMESSAGE page source.

This data update is shown in the solution in the IFrame.
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:

```javascript
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.

```javascript
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.

```javascript
function showbooks() {
    books = new DOMParser().parseFromString(books, 'text/xml');
    document.querySelector('code').innerText = books.childNodes[0].outerHTML;
}
```

The updated book list is shown in the webpage.

### 19.4.4 Pre-setting the JSON Page Source

The example in this section shows how to automatically send JSON source data from the webpage to the workflow when the HTML page is opened.

The HTML code is based on that used in the Sending/Receiving JSON Data example. 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

```html
<!DOCTYPE html>
<html>
<head>
<style>
* {
  font-family: Segoe UI, Tahoma, Arial, Helvetica, sans-serif;
}
iframe {
  width: 100%;
  height: 400px;
  border: 2px solid blue;
  border-radius: 5px;
  margin: 10px 0px;
}
code {
  font-size: small;
}
</style>
<script src="http://localhost:8083/js/WebAppIFrame.js"></script>

// 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
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):

<table>
<thead>
<tr>
<th>Color</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Webpage: user actions and how the HTML/JavaScript code works</td>
<td></td>
</tr>
<tr>
<td>Solution: actions carried out by the solution in the IFrame</td>
<td></td>
</tr>
<tr>
<td>Workflow: processing on the server (based on the MT design)</td>
<td></td>
</tr>
</tbody>
</table>

On loading the HTML webpage:
A JavaScript variable named `books` is read. It contains a JSON object named `books`.

```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
        }
    ]
};
```

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`):

```javascript
document.addEventListener('DOMContentLoaded', showbooks);

function showbooks() {
    document.querySelector('code').innerText = JSON.stringify(books, null, ' ');
}
```

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:

```javascript
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.

```html
<script src="http://localhost:8083/js/WebAppIFrame.js"></script>
```
**Examples**

**1281**

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:**

```javascript
'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 $SMT_EMBEDDEDMESSAGE$ page source.
The data has been made available in the IFrame directly on opening the HTML page.

19.4.5 JWT Authentication

The JWT authentication example in this section modifies the webpage of the Pre-setting page source example in the previous section. 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 g9khVQPKkNYts3CraUsmmF6RyEvTCFnt.
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.

```html
<!DOCTYPE html>
<html>
  <head>
    <style>
      * {
        font-family: Segoe UI, Tahoma, Arial, Helvetica, sans-serif;
      }
    
    iframe {
```
width: 100%;
height: 400px;
border: 2px solid blue;
border-radius: 5px;
margin: 10px 0px;
}
code {
    font-size: small;
}
</style>
<script src="http://localhost:8083/js/WebAppIFrame.js"></script>

// 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() {

```javascript
var embedded = new WebAppIFrame(document.querySelector('iframe'));

'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJPbmxpbmUgSldUIEJ1aWxkZXRlLCJpYXQiOjE1MDExMzU4MzUsImV4cCI6MTUzMzg1NjU1N30.
');
</script></head><body>
<h4>An editable list of books in JSON format</h4>
<p>The book list, stored as a JSON object in the webpage:</p>
<pre><code><!-- The SHOWBOOKS function displays the book list here --></code></pre>
<p>The book list is displayed in the Iframe as soon as the HTML page is opened.</p>
</body></html>
```
20 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 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](#)
20.1 Recording a Test Case

This action creates a test case, which can be used as the base case for comparisons.

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.
20.2 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. The playback is saved (whether there are differences or not).

Playback options

Options for recording and playback are available at the bottom of the Manage Test Cases and Runs dialog. 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:
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.

**Note:** Message boxes are not displayed during playback in MobileTogether Designer.

Playing back a test case on a client device

Playing back a test case on a client enables you to preview and check whether a given test case functions correctly on specific client devices. In this scenario, the MobileTogether Designer machine takes on the role of a MobileTogether Server. This means that you must set a separate port via which the Designer machine (acting as a server) will connect with the client.

The setup consists of two parts:

1. 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**.
2. 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. The playback is automatically saved (whether there are differences or not) as a test run when it is closed. To check whether there were differences, click Manage Test Cases and Runs in the Automated Testing toolbar.
20.3 Managing Test Cases and Runs

You can open the Manage Test Cases and Runs dialog by clicking the Manage Test Cases and Runs icon in the Automated Testing toolbar (see screenshot below).

In the Manage Test Cases and Runs dialog (screenshot below) you can do the following:

- Set up recording options for test cases.
- Set up recording and playback options for subsequent test runs.
- Specify that the test runs of individual test cases start with their persistent data reset to original values.
- Reload and save MobileTogether Recording files (.mtrecord files). At any given time, there is at most one .mtrecord file per design file, and it has the same name as the design file (but a different file extension).
- Delete and compare test runs.
- Substitute a test case with one of its test runs. The test run takes on the role of the test case. Other test runs are deleted, and the selected test run becomes the new test case of that (now empty) group.
- Deploy a test case to MobileTogether Server, retrieve test runs from the server, and delete a test case or test run from the server.

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 the test cases/runs and click Delete Selected. If you select a test case, then all its test runs will also be implicitly selected and deleted; however, you will be warned about this and prompted for a go-ahead. You can also delete all test runs across all test cases by clicking Delete All Playbacks.

If a test run is different than its test case, then the part that is different is indicated in the Compared pane by a pink button (see screenshot above). For example, if an action of a test run is different, then the test run’s Actions cell will contain a pink button (see screenshot above). If you click a pink button, the Compare Test Cases and Runs dialog 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). If you click Browse in this dialog, the deployed solutions are displayed together with the number of already deployed test cases in round brackets. (The number on angular brackets is the MobileTogether Designer version that was used to generate the test case.)

After a test case has been deployed to the server and been made active, it can be played back each time the solution is started on the client. If multiple test cases have been made active on the server, then all these test
cases will be played back on the client. Playback results are stored on the server. You can retrieve these playbacks to MobileTogether Designer by clicking **Retrieve** in the Manage Test Cases and Runs dialog (*screenshot above*). The retrieved playbacks are stored as test runs of the test case that was deployed, and can be handled in exactly the same way as other test runs.

You can delete deployed test cases from the server. Do this as follows: In the Manage Test Cases and Runs dialog (*screenshot above*), select the test case to delete, then click **Delete Deployed**.

For more details about these procedures, see [Deploying to Server](#).
20.4 Deploying Test Cases to Server

You can deploy one or more test cases of the active design to the server. If a test case is made active on the server, then it can be played back each time the solution is started on a client. In this way, a test case can be played back on multiple clients. These playbacks are stored on the server, and can be retrieved in MobileTogether Designer for comparison with the originating test case.

Deploying test cases to MobileTogether Server

A test case can be deployed with a design to MobileTogether Server via the File | Deploy to MobileTogether Server 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.

4. In the Security and Devices tabs, you can specify (i) the users/roles that are allowed to carry out test runs, and (ii) the devices on which to carry out test runs. Select the user/role or the device, and click Assign to allow the selected user/role or device.

If you wish to delete a test case, select its check box in the leftmost column and click Delete Selected. You can also delete a test case on the server by selecting it in the Manage Test Cases and Runs dialog (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 cases of that design, and does not entail the removal of test runs from the server. The retrieved test runs are displayed hierarchically under their originating test cases in MobileTogether Designer and can be handled in exactly the same way as every other test run. For example, you can compare a retrieved test run with its originating test case.
20.5 Comparing Test Runs

You can compare a test run with its originating test case to see differences in their steps, actions, page source data, styles, and client views (the layout coordinates of design components in specific clients). In the Manage Test Cases and Runs dialog, 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 green connector between the panes, with differences shown by dark red connectors (see screenshot above).

• Ignore specified page source or snapshot data nodes for differences. In the column between the two panes, right-click a difference connector and select **Ignore only <this node>** or **Ignore all <these nodes>** (see screenshot below). Ignored nodes are highlighted in yellow. These nodes will be ignored as differences in future playbacks. To consider differences in these nodes again, deselect **Ignore only <this node>** or **Ignore all <these nodes>**.

![Screenshot of comparison](image)

• The last item in both trees is the respective final state. If there is no difference between the final states, then the connector between them will be green; otherwise it will be red.
21 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.

<table>
<thead>
<tr>
<th>Solution on MobileTogether Client</th>
<th>AppStore App</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileTogether Designer project is deployed by developer to MobileTogether Server as a solution</td>
<td>MobileTogether Designer project is deployed by developer to MobileTogether Server as an AppStore App solution</td>
</tr>
<tr>
<td>MobileTogether Client is downloaded by end user from app store</td>
<td>AppStore App is downloaded by end user from app store</td>
</tr>
<tr>
<td>MobileTogether Client on end-user devices can access solutions on one or more MobileTogether Servers</td>
<td>AppStore App accesses its associated solution. It is &quot;dedicated&quot; to this one solution</td>
</tr>
<tr>
<td>End user needs to configure and maintain MobileTogether Client. Access to multiple servers and solutions is possible</td>
<td>No MobileTogether Client required. AppStore App provides end user with easy and straightforward access to a single solution</td>
</tr>
<tr>
<td>From MobileTogether Server, a deployed solution can be opened to run in a web browser.</td>
<td>The deployed AppStore App cannot be opened in a web browser from MobileTogether Server.</td>
</tr>
</tbody>
</table>
21.1 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
General

Build Modes

- Publish
  The packages generated by this build mode are intended for final release. The solution will be deployed on the server, and the packages can be made available in App Stores for customers.

- Trial Run on Client
  The packages generated by this build mode can only be used for testing with a Designer as server. The solution will not be deployed on the server but will be loaded directly from the Designer. These packages should never be uploaded to App Stores for customers.

App

The name of the executable file, which must consider the different platform constraints (e.g. "MyProductApp")

Executable file name: MyFirstApp

The name visible on the Home screen on the client (e.g. "My Product App")

Visible name: My First App

App version number, must be integral due to AppStore limitations

Version: 1

App languages (only Windows App, Windows Phone)

The languages that the app supports in addition to English.

The app’s user interface, including error messages, will appear in these languages only. The Windows App/Phone Store will require you to provide a description in each chosen language.

This setting is independent of the solution localization defined in the Localization dialog – all languages you provide there remain available.

Supported languages: French, Japanese, German, Spanish

For starting App from URLs (optional)

The URL scheme for starting your app via a link, e.g. mobletogether in mobletogether://mt/run-solution, generated by XPath function mt-run-apppstoreapp-url

URL scheme: myappscheme

The URL host for starting your app via a link, e.g. mt in mobletogether://mt/run-solution, generated by XPath function mt-run-apppstoreapp-url

URL host: myfirstapp
• **Build modes:** (i) The *Publish* option generates the app for publication to app stores; (ii) The *Trial Run on Client* option generates a package that can be used to test the package by **simulating a trial run on the client device**. The compiled app will connect to MobileTogether Designer and use the design file in MobileTogether Designer. This means that you would be able to modify the design in (MobileTogether Designer) and directly test the effect of the modifications in a client simulation. Note that if the *Trial Run on Client* option is selected, then the **Deploy Workflow to Server** step will be skipped—since the app must not be deployed to the server.

• **Executable file name:** The name that is used internally to reference the code. You should use a name that has no spaces.

• **Visible name:** The name of the app that will be visible to the user.

• **Version:** The version number of the app. Must be an integer or a decimal number. For example: 1 or 1.0 or 1.1 or 1.2. 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.
- **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](#). Click the Open File icon to view the image file in the system’s default image viewer app.

- **Launcher icon:** The icon that is displayed in the Apps screen of the mobile device to launch the app. The maximum pixel size of icons is 200x200. Click the Open File icon to view the image file in the system’s default image viewer app.

- **Copyright and legal notice:** The text to be displayed in the mobile device.

3: Server: Server and Login Settings
In Screen 1 you can select (in the Build Modes option) whether the appstore app you are currently building is for publication or for simulations on a client device. If you are generating for publication, then specify the connection details of the MobileTogether Server to which the app will connect. If you are generating for simulation on a client, then specify the connection details of the MobileTogether Designer machine. The screenshot above shows Screen 3 when the build is for publication; this is why the first pane takes the properties of the Server rather than those of the Designer.

- **Server/Designer:** The IP address of the server on which the workflow will be deployed (publish mode) or of the designer that you want to use for trial runs (client-simulation mode).
- **Port:** The port on the server/designer via which the app can be accessed. The server's client device access port is set in MobileTogether Server. See the MobileTogether Server user manual for information. For trial runs, you can use the designer as the server. The designer's client device access port is set in the Trial Run on Client tab of the Options dialog. For the Port setting, enter the Server's or Designer's device access port.
- **SSL:** If you use SSL, this will need to be set up in MobileTogether Server. See the MobileTogether Server user manual for information.
- **Always use anonymous login:** Select this option if you want to allow end users to access the app without providing login details. Otherwise, end users will need a user name and password to log in. See the MobileTogether Server user manual for information about setting up login credentials.
4: Privileges: User and App privileges

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\MobileTogetherDesigner7\MobileTogetherSPL. This directory will be the default SPL template directory.

If you customize one or more SPL templates, create a copy of the SPL template directory (in which the customized files will be saved). Specify its location of this directory in the SPL template directory option of this screen (see screenshot above). The new SPL template directory must reproduce the directory structure of the original. The option Use custom SPL template directory enables you to switch quickly between the original SPL template directory and your custom SPL template directory. Select or deselect the option according to which templates you want to use (custom or original).

6: Push Notifications
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
Sets the target directory where the program code will be generated for the Android format. You must specify the package name for the Android package. You can also select icons for Android's round launcher and adaptive launcher, as well as the background color of the adaptive launcher. To preview the image files, click the respective image's Open File icon. The Google API Key field is enabled if the design contains a Geolocation Map control. A Google API Key is needed for enabling the features of the control. You can get an API key via your developer account on the Google Cloud Platform.

8: iOS
Sets the target directory where the program code will be generated for the iOS format. You must specify the Bundle ID prefix for iOS. Make sure that the Bundle ID prefix ends with the dot character `. For example, `com.altova.` has the required format. The Bundle ID is constructed by appending the app name you provided in Screen 1 to the Bundle ID prefix. If you have specified that the app should have iCloud access, then an iCloud Container ID is automatically generated and stored in a file called `<appname>.entitlements`. This file will be created automatically in the target directory of the program code. (See Compile Program Code: iOS for more information.) You can also set the background color of the iOS launcher icon.

9: 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. The Publisher ID is the GUID that’s assigned to your developer account (see Windows App | Requirements; your Windows Publisher ID can be found on your Account Summary page in the Dev Center; the link in the dialog takes you there). You can also select a background color for the splashscreen of the app. The Bing Authentication field is enabled if the design contains a Geolocation Map control. A Bing Authentication Key is needed for enabling the features of the control. You can get an authentication key after registering as a developer at the Bing Maps Dev Center.

After you complete entering the information required by the Generate Program Code Wizard, click Finish. The Deploy App to Server dialog appears (described in the next section). Note: If the build mode that was selected in Screen 1 was Trial Run on Client, then the Deploy Workflow to Server step will be skipped—since the app must not be deployed to the server.

After the settings of the app have been defined, you can compile the program code for the respective platforms.
21.2 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.
21.3 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

21.3.1 Android

When you complete the wizard, MobileTogether Designer creates an Android Studio project. You can use Android Studio to build this project into an Android app (which is a .apk file).

The Android Studio project

The Android Studio project is created in the directory you selected as the program code's destination folder (in Screen 6 of the Generate Program Code Wizard).

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 3.3, which is the version that is current at the time of writing (2019).

Open the project (the target directory) in Android Studio. The build will start automatically, and the result is shown in the Gradle Console. During the build process, you might receive the message: AAPT err [...] libpng warning: iCCP: Not recognizing known sRGB profile that has been edited. This can be safely ignored. The APK file is not created immediately. You must run the app in order to create the APK file.

Running and debugging the app

When the build is complete, the app can be run by issuing one of the following Android Studio commands: Run <app> or Debug <app>. The APK file is created in: app\build\outputs\apk\debug\<app>-debug.apk.

Releasing the Android app

In order to publish the app, you will need to create a signed APK. Do this as follows:

1. In Android Studio, issue the command Build | Generate Signed APK. This starts the Generate Signed APK Wizard.
2. Type in the keystore path. You might need to create a new keystore first.
3. Go through the wizard's screens and fill in the information that is required.
4. Click Finish to start generating the signed APK.

The signed APK file will be placed in: app\release\<app>-release.apk. (If Debug was selected, the app will be placed in app\debug\<app>-debug.apk.)

Note: The Generate Signed APK command can cause Android Studio to erroneously report lint errors in open code files. If this occurs, issuing the command File | Invalidate Caches / Restart might remove the errors. If you receive the message: AAPT err [...] libpng warning: iCCP: Not recognizing known sRGB profile that has been edited, you can safely ignore it.

21.3.2 iOS

The generated program code for iOS apps is an XCode project. It is saved in the directory you specified in Screen 6 of the Generate Program Code Wizard. 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:

- 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:

```
<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](https://developer.apple.com) and [Enabling CloudKit in your app](https://developer.apple.com).

### 21.3.3 Windows App

The Windows App format is for Windows touch devices (such as tablets running Windows RT) and PCs running Windows 8.1 or higher. The generated program code for Windows Apps is in C++. It is generated in the directory you specified in **Screen 7 of the Generate Program Code Wizard**. You can open the generated C++ project (`some-app-name.vcxproj`) in Visual Studio and build the project into a Windows App. The output of the build process includes a file with the `.XAP` extension. This is a zip file that contains all the files required by your app, and it is the file that you publish to the Windows App Store.

For additional information, see the `Readme.docx` file in the MobileTogether Designer program files folder: `C:\Program Files (x86)\Altova\MobileTogetherDesigner7\MobileTogetherSPL\WindowsApp`.

**Requirements**

In order to build, test and publish the Windows App from the generated program code, you will need the following:

- A Windows app developer account. You will need this to publish your app to the Windows Store. For details, see the [Microsoft information about opening an account](https://developer.microsoft.com).
- Visual Studio 2015 Update 3, Visual Studio 2017, and Visual Studio 2019. For each of these versions, the respective additional components listed below are also required.
- Windows 8.1 or higher for testing the app.

**Additional components for different Visual Studio versions:**

**Visual Studio 2015 Update 3**
- Universal Windows App Development Tools (1.4.1) and Windows 10 SDK (10.0.14393)
- Windows 8.1 and Windows Phone 8.0/8.1 Tools: Tools and Windows SDKs

**Visual Studio 2017**
• C++ Universal Windows Platform Tools
• Windows 10 SDK (10.0.14393.0)
• Windows 8.1 SDK (can be found under "SDKs, libraries, and frameworks" in "Individual components")

Visual Studio 2019
Either one of the following:
• Visual Studio has the following components: (i) Windows 10 SDK (10.0.17134.0); (ii) MSVC v141 – VS 2017 C++ x64/x86 build tools; (iii) MSVC v141 – VS 2017 C++ ARM build tools;
• The generated project is re-targeted to use Toolset v142. Do this as follows: In Visual Studio’s Solution Explorer: (i) select the project; (ii) run the Retarget Projects command in the context menu. If you expect your app to be used on Windows Phone, then select Windows SDK 10.0.17134.0 or older. Note that version 10.0.17763.0 or newer will generate a .msix installer package, which cannot be used on Windows Phone. Note also that the minimum version of the Windows SDK that the MobileTogether libraries support is 10.0.10586.0.

Building the app
The broad outline for building the app is as follows:

1. Open the generated program code in Visual Studio. The file to open will be a .vcxproj file in the target directory you specified in Screen 7 of the Generate Program Code Wizard.
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.
21.4 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, and Windows.

- The SPL templates are delivered with your installation of MobileTogether Designer and are located in the folder C:\Program Files (x86)\Altova\MobileTogetherDesigner7\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\MobileTogetherDesigner7\MobileTogetherSPL

This folder contains entry-point SPL templates for each app format (Android.spl, iOS.spl, WindowsApp.spl, and WindowsPhone.spl), a common library SPL template (CommonLibrary.spl, which must not be modified), and a folder for each app format that contains additional SPL templates for that format. Inside the folder for each app format is a ZIP file and the SPL template files for that app format.

Each SPL file is a template that can generate a number of project files and call other SPL files. When creating an Android Studio project, for example, MobileTogether Designer begins by processing Android.spl (in the SPL template directory). This SPL file in turn calls other SPL files, which then generate other project files and call other SPL files. These SPL template files are the files that you can modify to alter the generated program code. They are written using SPL syntax and SPL instructions, which are described in the sub-sections of this section.

Using the SPL templates to modify program code

The steps to customize generated program code are as follows:

1. Copy either the entire SPL template directory (or only the templates you wish to modify) to a directory of your choice. In this way a re-installation of MobileTogether Designer will not overwrite your customized SPL templates. Note that the copy of the directory must have the same structure as the original directory.
2. Modify the SPL files as required.
3. Run the Generate Program Code Wizard 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:
21.4.1 SPL Syntax

An SPL template is constructed in the programming language of the program code you wish to generate. The template contains snippets of SPL instructions to integrate MobileTogether data into the generated program code. SPL instructions are enclosed in square brackets. Here, for example, is a template to generate an XML file (written in XML), with the SPL instructions highlighted in yellow.

```xml
<?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]
```

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:

```spl
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.

```spl
if condition
    statements
else
    statements
endif
```

**Example**

```spl
[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:

```spl
foreach object in collection
    statements
next
```
Example

[foreach $class in $classes
  if not $class.IsInternal
] class [=$class.Name];
[ endif
next]

21.4.2 String Mechanisms

SPL provides the string manipulation mechanisms that are listed below. These methods are applied on the input string itself.

integer Compare(s)
The return value indicates the lexicographic relation of the string to s (case-sensitive):
  <0    the string is less than s
  0     the string is identical to s
  >0    the string is greater than s

integer CompareNoCase(s)
The return value indicates the lexicographic relation of the string to s (case-insensitive):
  <0    the string is less than s
  0     the string is identical to s
  >0    the string is greater than s

integer Find(s)
Searches the string for the first match of a substring s. Returns the zero-based index of the first character of s or -1 if s is not found.

string Left(n)
Returns the first n characters of the string.

integer Length()
Returns the length of the string.

string MakeUpper()
Returns a string converted to upper case.

string MakeUpper(n)
Returns a string, optionally with the first $n$ characters converted to upper case.

```csharp
string MakeLower()
```
Returs a string converted to lower case.

```csharp
string MakeLower(n)
```
Returns a string, optionally with the first $n$ characters converted to lower case.

```csharp
string Mid(n)
```
Returns a string starting with the zero-based index position $n$.

```csharp
string Mid(n,m)
```
Returns a string starting with the zero-based index position $n$ and the length $m$.

```csharp
string RemoveLeft(s)
```
Returns a string excluding the substring $s$ if `Left(s.Length())` is equal to substring $s$.

```csharp
string RemoveLeftNoCase(s)
```
Returns a string excluding the substring $s$ if `Left(s.Length())` is equal to substring $s$ (case-insensitive).

```csharp
string RemoveRight(s)
```
Returns a string excluding the substring $s$ if `Right(s.Length())` is equal to substring $s$.

```csharp
string RemoveRightNoCase(s)
```
Returns a string excluding the substring $s$ if `Right(s.Length())` is equal to substring $s$ (case insensitive).

```csharp
string Repeat(s,n)
```
Returns a string containing substring $s$ repeated $n$ times.

```csharp
string Replace(sOld,sNew)
```
Replaces the string $sOld$ with the string $sNew$.

```csharp
string Right(n)
```
Returns the last $n$ characters of the string.

```csharp
string TrimLeft()
```
Returns the string after trimming spaces from the left. Trimming stops when a non-space character is encountered.

```csharp
string TrimLeft(s)
```
Returns the string after removing, from the left, all the characters contained in `s`. Trimming stops when a character is encountered that is not contained in `s`.

### string TrimLeftRight()
Returns the string after trimming spaces from the left and the right. Trimming stops when a non-space character is encountered.

### string TrimLeftRight(s)
Returns the string after removing, from the left and the right, all the characters contained in `s`. Trimming stops at left and at right when a character is encountered that is not contained in `s`.

### string TrimRight()
Returns the string after trimming spaces from the right. Trimming stops when a non-space character is encountered.

### string TrimRight(s)
Returns the string after removing, from the right, all the characters contained in `s`. Trimming stops when a character is encountered that is not contained in `s`.

**String properties**
The following properties are available:

- **Length**: returns the length of the string. **Example**: `$Options.deploymentPath.Length` returns the length of the string contained in `deploymentPath`.
- **XMLEncode**: returns the length of the string in XML-encoded format. **Example**: `$Options.deploymentPath.XMLEncode` returns the string contained in `deploymentPath` as XML-escaped text.

### 21.4.2.1 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.

```xml
<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](#).

- `isTrialRunOnClientBuild`
- `appName`
- `visibleAppName`
- `appVersion`
- `hostForRunSolutionUrl`
- `schemeForRunSolutionUrl`

User interface properties
The values of these properties are provided in [Screen 2 of the Generate Program Code Wizard](#).

- `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 7 of the Generate Program Code Wizard.

targetDirectoryAndroid
androidPackageName
androidPackageDir
androidRoundLauncherIconFilePath
androidAdaptiveLauncherIconForegroundFilePath
isAndroidAdaptiveLauncherIconBackgroundColor
androidAdaptiveLauncherIconBackgroundFilePath
androidAdaptiveLauncherIconBackgroundColor
targetDirectoryIOS
iosBundleIdentifierPrefix
iosLauncherIconBackgroundColor

Windows App properties
The values of these properties are provided in Screen 8 of the Generate Program Code Wizard.

windowsAppCompanyName
windowsPhoneCompanyName
windowsCompanyPublisherID
targetDirectoryWindowsApp
windowsAppCompanyProductID
targetDirectoryWindowsPhone
windowsPhoneCompanyProductID

21.4.3  Properties of $Application

The $Application object can take the properties listed below. The object's properties can be accessed by
using the . operator; see the Examples section below.

Properties

CopyFile(sSource, sTarget)
Copies the file at the location sSource to the location sTarget.

UnzipFile(sZipFile, sTargetDir)
Unzips the ZIP archive at the location sZipFile to the directory sTargetDir.

FileExists(sFile)
Checks whether the file at location sFile exists. Returns 1 or 0.
DeleteFile(sFile)
Deletes the file at the location sFile.

WriteDesignFileContent(sContentFile)
On code-generation, writes the content of the design file to the app-package using the file specified by sContentFile. (Deployed client files will also be written to the package.)

CopyAndResizeImage(sSource, sTarget, nTargetWidth, nTargetHeight)
Copies the image file at the location sSource to the location sTarget, and resizes the copied image to the pixel dimensions given by nTargetWidth and nTargetHeight.

Examples
Here are some SPL template usage examples:

```
sub CopyFile( byval $sSource, byval $sTarget )
  return $Application.CopyFile( $sSource, $sTarget )
endsub

sub UnzipFile( byval $sZipFile, byval $sTargetDir )
  return $Application.UnzipFile( $sZipFile, $sTargetDir )
endsub
```

21.4.4 Miscellaneous Objects

The following objects can be accessed as variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Host</td>
<td>MobileTogetherDesigner &lt;version, e.g. 2.0&gt;</td>
</tr>
<tr>
<td>$HostShort</td>
<td>MobileTogetherDesigner</td>
</tr>
<tr>
<td>$HostURL</td>
<td><a href="http://www.altova.com/mobiletogther">http://www.altova.com/mobiletogther</a></td>
</tr>
<tr>
<td>$HostVersion</td>
<td>Major product version (for example 1 when 1.5)</td>
</tr>
<tr>
<td>$HostRelease</td>
<td>Minor product version (for example 5 when 1.5)</td>
</tr>
<tr>
<td>$RegisteredName</td>
<td>Name of the licensed user</td>
</tr>
<tr>
<td>$RegisteredCompany</td>
<td>Licensed company</td>
</tr>
<tr>
<td>$CreationDate</td>
<td>Time of SPL code generation</td>
</tr>
<tr>
<td>$outputpath</td>
<td>Directory where the code is generated</td>
</tr>
</tbody>
</table>
22 In-App Purchases

In-app purchases are content or subscriptions that you buy from inside an app on a mobile device. For example, you can buy coins from inside a game app or a subscription from inside a music app. The coins and the subscription are in-app purchases—because they have been purchased from inside an app.

MobileTogether Designer enables you to add in-app purchases as a feature of your AppStore Apps.

In this section, we describe the mechanisms to add the in-app purchase functionality to the MobileTogether design of your AppStore App. After adding in-app purchases to your design, you would generate your AppStore App (from the design) for one or more of the supported platforms (Android, iOS, Windows). Note that the in-app purchase functionality only works in AppStore Apps; it will not work in MobileTogether solutions that run in MobileTogether Client—because of app-store restrictions.

You must of course make sure that the in-app purchases—the products that you want end users to buy—have been registered with the respective app store/s and are available for sale there. The end user of your AppStore App can now buy from the app store. In your MobileTogether design, you will design the interactions between your AppStore App and the app store so as to correctly transact an in-app purchase.

How it works

• Generate your AppStore App from your MobileTogether Designer project and upload it to the respective app store.
• In the app store, make available your products for in-app purchases.
• After the AppStore App has been downloaded to a mobile device, the end user can carry out in-app purchases.

Below, we first provide an overview of the categories of in-app purchases and then outline the broad steps for setting up in-app purchases in a MobileTogether design. In the sub-sections of this section, we describe these steps in detail.
App store documentation

For information about the procedures at the different app stores, see the relevant documentation at the respective app store. It is important that you understand the workflow requirements for each app store and make sure that you implement them correctly when creating the design of your AppStore App.

The links below provide entry points to the app store documentation.

- Android/Google: https://developer.android.com/google/play/billing
- iOS/Apple: https://developer.apple.com/documentation/storekit/in-app_purchase
- Windows/Microsoft: https://docs.microsoft.com/en-us/windows/uwp/monetize/
- Windows/Microsoft: https://docs.microsoft.com/en-us/windows/uwp/publish/add-on-submissions

Categories of in-app purchases

There are three categories of in-app purchases:

- **Consumables**: These are purchases that are consumed within the app, such as game currency. They are bought, depleted, and can be purchased again.
- **Non-consumables**: These are content or services that are purchased once and do not expire, for example, additional filters in a photo app.
- **Subscriptions**: These are access to periodically updated content or services, such as a monthly magazine. Subscription charges recur until canceled.

Setting up in-app purchases for an MT project

The broad steps for enabling in-app purchases in your AppStore App are as follows:

1. Via your app store account, create the in-app products (or SKUs, stock-keeping units) that you want to make available as in-app purchases. During this process, you will give each product, among other properties, an ID, a name and description, and set the product's price. Product IDs in the different app stores each have a specific and different format. So note down these product IDs; you will need them in the MT design when you map the store IDs to the product's name in the design. In the design, you will work with this single product name.

2. In the MT design, for each product that you want to make available as an in-app purchase, map that product's IDs in the different app stores to one product name. This product name will be used in the design as the product's identifier.

3. Information about products is stored at the app store. This information can be retrieved by MobileTogether's Query Available Products action, and it is stored in the design's In-App-Purchase Page Source.

4. Information about the current user's purchases is also stored at the app store. It can be retrieved by MobileTogether actions and is also stored in the design's In-App-Purchase Page Source.

5. The product and purchase information that is available in the In-App-Purchase Page Source can be displayed in the design by using suitable controls.

6. To make an in-app purchase, the end user triggers the Purchase action, which identifies the product to be purchased and the current user.

7. The app store executes the purchase transaction and sends transaction data to the app on the mobile device.

8. In the app, the purchased product (for example, a subscription) is activated (by you, the designer).

9. The app acknowledges the purchase, and this completes the transaction.
22.1 Register Products

The products that you want to sell in your app must be:

- registered and available in the respective app store (Google, Apple, Windows), and
- registered in the MobileTogether design under a single product name; essentially, the respective IDs of the product in the different app stores are mapped to a single name that serves as the product's ID in the design (see the screenshot below).

During the registration (or creation) of the product, you will have given the product an ID and set its price (among other things). Since each app store specifies a different format for the ID of products (compare the SKU ID formats in the screenshot below), the different IDs are mapped to a single product name (in the Product column of the dialog shown below). In the MT design, a product is identified by the name given to it in this dialog. When the respective AppStore Apps are generated from the design, the appropriate SKU ID is used for each platform. This mapping mechanism enables all design procedures related to a single product to be handled in a unified way in the design regardless of platform—by using the product's name, as defined here.

To set up the ID-to-Product-name mapping, do the following:

1. In MobileTogether Designer, select Project | In-App Purchase Products.
2. In the In-App Purchase Products dialog that appears (screenshot above), enter a name for each product that you want to make available for purchase.
3. In the respective SKU columns, enter the respective product IDs that you assigned to the product when the product was registered at the respective app store.

Click the Plus icon to add a new product. Click the Delete symbol to remove a product. After you have entered the data for all the products you want to make available for in-app purchases, click Close.

Also see how to create an in-app product for Android and how to create a subscription at the Google Play Store.

XPath functions for retrieving product name and ID
MobileTogether Designer provides two functions related to the naming mechanism: for retrieving (on a given platform) the product name from the product ID, and vice versa. You can use these functions at any location and at any time in the workflow.
Retrieve platform-specific product ID from product name
The `mt-in-app-purchase-product-to-platform()` function takes as its single argument a product name as entered in the In-App Purchase Products dialog (see screenshot above). It returns the ID of the product on the current platform. In simulations, the current platform is the platform of the simulation device. For example,

```
```

Retrieve product name from platform-specific product ID
The `mt-in-app-purchase-platform-to-product()` function takes as its single argument a product ID as defined for the current platform in the In-App Purchase Products dialog (see screenshot above). It returns the product name of the submitted product ID. In simulations, the current platform is the platform of the simulation device. For example,

```
mt-in-app-purchase-platform-to-product("3_Subscription_1M") returns "03-Subscription1M" on Android devices.
```

See MobileTogether Extension Functions for a complete listing and description of MT functions.
In-App-Purchase Page Source

Data about your products that are available for in-app purchases, as well as data about purchases made by an end user, are stored in the respective app stores.

When this data is needed on the client device for an in-app purchase, it is requested by the client from the app store and downloaded to a special page source in your MT design: `$MT_IN_APP_PURCHASE` (see screenshot below). During the design phase, this page source is added automatically to the design the first time you add any of the In-App Purchase actions.

The `Root` element of the page source contains two elements:

- **Purchases** contains data about individual purchases, each of which is stored in a child `Purchase` element (see tree in screenshot below). The attributes of the `Purchase` element correspond to the data points of a purchase at the app stores. While the actual data points are different (in number and name) from store to store, they are all mapped to the attributes of the `Purchase` element. This enables the design to reference a common set of abstract data points that can be used across all platforms. Here is an example of how the `Purchase` element is used: The `Query Purchases` action gets data about one or more purchases and places the data of each purchase in a separate `Purchase` element (see screenshot below left).

- **Products** contains data about products that are available for in-app purchases. Data pertaining to each product is stored in a separate child `Product` element. Just as with purchases, while the actual data items relating to a product are different from store to store, this data is stored in a single set of attributes of the `Product` element. Here is an example of how the `Product` element is used: The `Query Available Products` action returns data about the queried product/s and places the data of each product in a separate `Product` element (see screenshot below right).
Note the following points:

- While SKU_ID is an attribute of the Product element, it is a child element of the Purchase element.
- Purchase data from an app store is typically sent as a JSON string. MobileTogether retrieves key datapoints from this purchase data and saves them, for each purchase, in the attributes of a Purchase element. On Android and Windows systems, the original JSON string containing the data is stored in an @OriginalJSON attribute of the Purchase element, whereas on iOS systems the original JSON string is stored as an attribute of the Purchases element. This is because the Apple Store, when queried for the purchase data of an end user, returns data of all purchases rather than the data of each purchase separately (see the description of the Restore Purchases action). For a description of how to query purchases from the Google and Windows app stores, see the topic Query Purchases.
22.3 Query Available Products

In order for an end user to be able to buy a product, the products that are available for in-app purchase must be listed in the app. The mechanism to obtain a list of available products works as follows:

1. The Query Available Products action queries the app store for the details of a set of products. These products are identified in the design by their names, which are submitted as a sequence of strings (see screenshot below).

2. The app store returns the data about the queried products, and this data is stored in the $MT_IN_APP_PURCHASE page source. Each product's details is stored in a Product child element of Products (see screenshot below).

3. Since the product data is now in a page source, any data from the page source tree can be now displayed in the design. A suitable way to display all available products would be via a dynamic table, with each product being displayed in a row group of the table. You do not need to display all details of a product, but can select only those data points that you want to display: for example, the
product name, its description, and its price. Run a simulation of the example project to see how the mechanism to query available products works.

**Note:** In the Query Available Products action, if no specific product is specified, then all the products defined in the In-App Purchase Products dialog are queried.
22.4 Purchase Products

The core procedure of an in-app process is the transaction that carries out the actual purchase. It is implemented via the Purchase action, and it works as follows:

1. A Purchase action sends a purchase request from the app to the app store. The request contains (i) the name of the product to be purchased, and (ii) details of the user account to be used for the transaction (see screenshot below).

2. The app store responds with a summary of the product's key data and asks for confirmation of the purchase. The end user on the client device confirms the purchase.

3. The app store carries out the transaction and sends data about the transaction to the app. This data about the purchase is stored in the $MT_IN_APP_PURCHASE page source, inside a Purchase element (see screenshot below).

4. Once the purchase data has been received in the design, the actions of the OnPurchaseUpdated event are triggered. These are actions that you define. Typically, you would carry out the following steps:
i. **Check the response code** that the app store sends about the success/failure of the purchase request. You can retrieve the response code by using the function `mt-last-in-app-purchase-response-code()`. If the purchase request was successful, the function will return 0. Other return codes vary across platforms. (See the topic Other Operations for additional diagnostic functions, such as to check whether the user canceled the purchase.)

ii. If needed, save the purchase state of the product outside the `$MT_IN_APP_PURCHASE` page source (either in `$PERSISTENT`, some other page source, or an external file). You might want to do this if you want to keep, for some particular reason, a separate history of purchase activity.

iii. Activate the purchased product in the app. After checking the purchase state, subscription validity, possible cancellation, etc, of the purchase, you will need to activate the product accordingly.

iv. Acknowledge the purchase (via the **Acknowledge Purchase** action). You, as the designer, must trigger the **Acknowledge Purchase** action in order to finalize the purchase so that the vendor of the app (which could be you or some other entity) receives payment. Note that acknowledgment may be one of two types: (i) for subscriptions and non-consumable products, and (ii) for consumable products. For details, see the **Acknowledge Purchase** action and Example Project: The OnPurchaseUpdated Event. You should acknowledge the purchase after you have activated the product (see previous step); this would ensure that the end user receives the use of the purchase before any data related to the purchase is possibly modified.

Once a purchase has been acknowledged, the in-app purchase will be complete. Afterwards, you can query purchases and display information about purchases by using the **Query Purchases** or **Restore Purchases** actions. In the case of consumables that have been purchased for Windows devices, you can also use the **Get/Report Consumable** action to query how many credit units are still available (balance) and how many have been consumed (fulfilled).
22.5 Other Operations

The preceding topics of this section described the main components of MobileTogether Designer’s In-App Purchases mechanism. In this topic, we list and briefly describe the other key components of the mechanism.

Check availability of In-app Purchases service
You can use a function named `mt-in-app-purchase-service-started()` to check whether the in-app purchase service has been started on the client device. For details, see the topics MobileTogether Extension Functions and Example Project: Availability of In-App Service.

Check app store response to last in-app purchase
The following MobileTogether extension functions can be used to check the success of the last request to the app store that was related to an in-app purchase:

- `mt-last-in-app-purchase-response-code()`
- `mt-last-in-app-purchase-response-text()`
- `mt-last-in-app-purchase-response-was-user-canceled()`

This enables you to make the next step in the workflow dependent on the success of the current step. For an example, see the actions of the OnPurchaseUpdated Event in the Example Project. Note that these functions can be used to obtain the response of the app store to the last request made to it that was related to an in-app purchase.

An app store might respond with any of a number of responses, depending on which app store is involved. A code of 0 indicates a successful operation. The other codes indicate different types of failure. The function that returns response text would provide more information about the failure. One type of failure, however, should be handled differently since it is not an error. This would be when the end user cancels a purchase. To distinguish this type of purchase failure from errors, MobileTogether provides the function `mt-last-in-app-purchase-response-was-user-canceled`.

OnPurchaseUpdated: when an app store accepts a purchase request
When an app store accepts a purchase request, the purchase is automatically added to the In-App-Purchase Page Source. The purchase is said to be updated (in the page source) and the OnPurchaseUpdated event is triggered. A sequence of actions can be defined for the event, which enables you to seamlessly carry out actions, without requiring any user input, after a purchase request has been accepted. For example, you might want to acknowledge a purchase directly after it has been accepted (see next topic below). To access the event's Actions dialog, go to the project property In-App Purchase Actions.

Note the following points about the actions of the OnPurchaseUpdated event:

- While the actions of the OnPurchaseUpdated event are being processed, the `$MTUpdatedInAppPurchases` variable is in scope. The variable contains a sequence of strings, which are the SKU IDs of purchased products that have been updated in the In-App-Purchase Page Source. So, if one item has been purchased, then its SKU ID can be obtained with the XPath expression `$MTUpdatedInAppPurchases[1]`. The variable goes out of scope once the event's actions have been processed. This means that the variable can only be used in actions of the OnPurchaseUpdated event.
• In the In-App-Purchase Page Source, each Purchase element has a PurchaseState attribute that takes a value of either PENDING or PURCHASED. You might want to suitably modify your workflow to take this status into account.

For an example (i) of the kind of actions that can be defined for the OnPurchaseUpdated event, and (ii) of how to use the $MT_UpdatedInAppPurchases variable, see the description of this event in the example project: The OnPurchaseUpdated Event.

Activate product
Activate the purchased product in the app. After checking the purchase state, subscription validity, possible cancellation, etc, of the purchase, you will need to activate the product accordingly. This is a step that you as the designer of the app must carry out; MobileTogether cannot automatically do this for you.

Acknowledge Purchase
After the end user successfully buys a product via an in-app purchase, the app store accepts the request and sends information about the purchase to the app. On receiving this information, the purchase must be acknowledged from the app side, and this is something that you, as the app designer, must do silently without bringing the end user into the picture. The acknowledgment step enables the app to process the information received by the app store to determine whether the purchase is legitimate and whether to close the purchase from the vendor side. This provides you, as the app designer, with an opportunity to verify different aspects of the purchase. The app stores provide documentation that give you detailed information about what you should assess before acknowledging the purchase. Kindly read the relevant app store information before deciding on the steps to take before acknowledging. It is only after you acknowledge the purchase, that the app store will process the payment and route it to the product-owner's account.

Purchases are acknowledged via MobileTogether's Acknowledge Purchase action, and it is best to include this action as one of those to be carried out when the OnPurchaseUpdated event is triggered (see previous section above).

For details of this action, see the description of the Acknowledge Purchase action and the description of its implementation in the example project.

Query stores for purchase data
You can query the app store for all the purchases transacted by the current user. The action to do this is the Query Purchases action. When the action is executed, the app store returns data about the user's purchases, and key datapoints are stored in the Purchases element of the In-App-Purchase Page Source. Each purchase is stored in an individual Purchase element. You can then use this purchase data in the app's workflow. For detailed information, see the description of the Query Purchases action and the description of the Query Purchases button in the example project.

Restore purchases (iOS only)
The Apple Store keeps a record of all of an end user's purchases and enables the end user to restore all these purchases on an iOS device at any time. The Restore Purchases action can be used to query the Apple Store for all of an end user's purchases. The data returned is stored in the Purchases element of the In-App-Purchase Page Source, with each purchase being stored in an individual Purchase element. You can then use this purchase data in the app's workflow. For detailed information, see the description of the Restore Purchases action and the description of the Restore Purchases button in the example project.
Get/Report Consumables (Windows only)

After a consumable has been purchased, it can be consumed within the app. In the case of Windows apps, the Windows app store offers an option to keep a record of a user's current consumable balance. The Get/Report Consumable action addresses this Windows option. It can get the current balance from the Windows app store and report the fulfillment level of the consumable to the Windows app store. For details see the description of the action.

Note: It is your responsibility to check the current level of consumable credit, especially in the case of iOS and Android, where the respective app stores—unlike the Windows app store—does not offer an option for tracking consumable usage.

Keep track of the product's activation status

It is your responsibility to determine whether a product is active within the app at any given time. This decision would be based on the following factors:

- The product's PurchaseState as reported by the app store.
- In the case of consumable products, the current level of consumable credit. You will have to keep track of this in your app.
- In the case of subscriptions, the current state of the subscription relative to the start date and period of the subscription.
- Any other factor that affects the activation status of the product, such as a free extension period for a subscription.

REST interfaces

You can also use REST interfaces as far as these are supported on the respective platforms. The use of REST interfaces with AppStore Apps on Android has been tested, and the documentation of these interfaces is available here: https://developers.google.com/android-publisher/api-ref/rest.
22.6 Simulation and Testing

You can test the in-app purchases of your AppStore App by simulating it in two easy ways, as described below.

Simulation in the Designer
This simulation enables you to test the workflow with product and purchase data that is stored in an XML file.

This XML file is located by default at "<Program Files>\Altova\MobileTogetherDesigner7\InAppPurchase\InAppPurchase Samples.xml". Make sure that you map the SKU IDs in this file to a product name as described in the topic Register Products. Note that you can save your XML data file at any other location if you like; in this case, change the reference to the file in the Simulation 2 tab of the Options dialog.

Note that the simulation in MobileTogether Designer works with static product and purchase data, which is taken from the XML file listed above. While a designer simulation broadly simulates the actual procedure, a trial run on client simulation (see below) more closely simulates the various steps of the live procedure.

Simulating trial run on client
While a simulation in the designer that uses dummy data is useful for testing the workflow steps on a broad level, a significant problem with Designer simulations is that you cannot use real app store data for the simulation. A better way to test your AppStore App is to generate a Trial Run build of the app that can directly use the design that you are creating in MobileTogether Designer. To generate a Trial Run build, make sure that the following two settings are correctly selected when you build the app:

- In Screen 1, in the Build Modes pane, select Trial Run on Client.
- In Screen 3, enter the correct IP Address and Port settings so that the app can connect to your MobileTogether Designer.

Once the Trial Run build has been generated, it will connect to the design in MobileTogether Designer and use the latest version of the design. This saves you the trouble of having to regenerate the AppStore App to test it each time you make a change to the design. Once you are satisfied with your tests and ready to publish, you can generate a final publication build that you can upload to the app store.
22.7 Example Project

An example project named `InAppPurchases.mtd` is located in your *(My) Documents* folder `Altova\MobileTogetherDesigner7\MobileTogetherDesignerExamples\Tutorials\InAppPurchases`. The project implements a simple in-app purchase workflow to demonstrate the MobileTogether mechanism for in-app purchases. For you to get a broad understanding of how to set up in-app purchases from your project, we recommend that you run a simulation of the project in MobileTogether Designer and then inspect the settings of various project components. The topics of this section describe the key components of the example project.
Since the simulation in MobileTogether Designer does not use a real account or connect to an app store, it uses data in the file `InAppPurchase_Samples.xml` to simulate products and purchases. This data file is located in your Program Files folder `Altova\MobileTogetherDesignerX\InAppPurchase`. This file has already been set as the default file to use for the simulation of in-app purchases, in the `Simulation 2 tab` of the Options dialog (`Tools | Options`).
Note: Since the simulator uses the device selected in the Preview Device combo box, it will simulate purchases at the corresponding app store. To simulate purchases at a different app store, change the preview device suitably.

The data in the file "InAppPurchases-Data.xml"
The broad structure of the data file is this:

```
Root
|--- Android
|   |--- Products
|   |   |--- Product
|   |--- Purchases
|   |   |--- Purchase
--- iOS
|   |--- Products
|   |   |--- Product
|   |--- Purchases
|   |   |--- Purchase
--- Windows
|   |--- Products
|   |--- Purchases
|   |   |--- Purchase
```

The data file contains four products and four purchases (for each platform/store). Note that the data in this file will be selected according to which device is selected for the simulation in the Preview Device combo box. For example, if an Android device has been selected as the Preview Device, then data inside the Android element will be used for the simulation.

22.7.1 Map Product IDs to Product Name

Each product will have a different SKU ID at each app store. In the In-App Purchase Products dialog, for each product, map the three SKU IDs to a single product name. This name will be used to identify the product in the design.

The SKU IDs are taken from the data file you use in simulations (see above).
22.7.2 The $PERSISTENT Page Source

The $PERSISTENT page source has three elements (see pane at right in screenshot below).

- `ServiceAvailable`: which holds text that indicates *whether the In-App Purchases service is available* on the device
- `ProductCount`: which holds the count of available products
- `PurchaseCount`: which holds the count of purchases

The values of these elements are each displayed in a label, done by setting that element as the page source link of the label. Each label is next to a button in the blue table. When a button is clicked, the corresponding node in the $PERSISTENT page source is updated, and the updated node value is displayed in the label corresponding to that button.

22.7.3 Availability of In-App Service

Each operating system (Android, iOS, Windows) offers a service that sets up the device for in-app purchases at the corresponding app store. The MobileTogether function `mt-in-app-purchase-service-started()` checks whether the service has been started on the respective operating system. It returns `true()` or `false()`.

In the example project, this functionality has been enabled as a button-click action on the button *Is InApp Service Available?* (see screenshot below).
When the button is clicked, the function `mt-in-app-purchase-service-started()` is called and the `ServiceAvailable` node in the $PERSISTENT tree is updated based on the value returned by the function. Since this node is the page source link of the label on the right-hand side of the button, the node’s value is immediately displayed in the label.

### 22.7.4 Get Available Products

On clicking the button *Get Available Products*, the [Query Available Products](#) action is executed. At run time, the action would get the products from the app store. In our simulation, the action gets the products from the [simulation data file](#).

In our example project, the action is defined with the `Product IDs` setting empty ([see screenshot below](#)). As a result, all the products defined in the [In-App Purchase Products](#) dialog are queried ([see Step 1 above](#)).

After the products are queried, the app store sends back data about the queried products to the client device and this data is stored in the `$MT_IN-APP_PURCHASE` page source ([see Query Available Products](#) for details). The actions to send the request and store the data are carried out automatically by MobileTogether.

As part of the design of our example project, we have implemented two additional steps:

- The button *Get Available Products* specifies an [Update Node](#) action, which counts the number of products now in the page source tree and updates the `$PERSISTENT/Root/ProductCount` node with...
this number. Since this node is the page source link of the label corresponding to the Get Available Products button, the number of products shown in this label is updated immediately.

- The products that are now stored in the page source tree are displayed in a dynamic table, with each product being displayed in a row group of the table (see screenshot below).

The effect of both the actions listed above can be seen in the simulator screenshot shown below. When the button Get Available Products button is clicked, the four products that are stored in the XML data file are queried (in the case of an Android device simulation, these are the Product elements of the Android element). This product data is sent to the device and stored in the Product elements of the $MT_IN_APP_PURCHASE page source.
After the two additional steps mentioned above have been carried out and the $MT_IN_APP_PURCHASE page source has been updated with the new Product data, the following happens:

- The count of products is updated, and
- Information about each product is displayed in a row of a dynamic table.

**Note:** The green label above the Available products table has its Visible property set to the XPath expression $MT_IN_APP_PURCHASE/Root/Products/Product. This causes the label to be visible only when there is at least one Products/Product element.

**Note:** The table itself will be visible only when it has content; so there is no need to set a condition for visibility.

In the next section, we describe the Buy button that is displayed with each product.
22.7.5 The "Buy" Button

In our example project, the end user can get a list of products for in-app purchase by tapping the Get Available Products button (see screenshot below left). For each product in this list, there is a Buy button that the end user can use to buy the product (screenshot below left). The screenshot below right shows the design, in which the button can be seen at bottom right. The button has two important attributes: (i) its text, (ii) the purchase request that is sent to the app store.

These are described below specifically for Android devices. You might need to modify the design to suit a different app store's requirements.
In-App Purchases

Is InApp Service Available? Available
Get Available Products 4 products available
Query Purchases No purchases yet
Clear Page Source Tree

Available products

<table>
<thead>
<tr>
<th>SKU_ID</th>
<th>Description</th>
<th>Price</th>
<th>Buy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1_Consumable_Product</td>
<td>A consumable test product</td>
<td>€ 3.59</td>
<td></td>
</tr>
<tr>
<td>2_NonConsumable_Product</td>
<td>A one-time purchase</td>
<td>€ 2.39</td>
<td></td>
</tr>
<tr>
<td>3_Subscription_1M</td>
<td>Monthly subscription</td>
<td>€ 5.99</td>
<td></td>
</tr>
<tr>
<td>4_Subscription_1Y</td>
<td>Yearly subscription</td>
<td>€ 24.00</td>
<td></td>
</tr>
</tbody>
</table>

Go to top
Since the **Buy** button is placed in a row group that repeats for each **Product** element in the `$MT_IN-APP_PURCHASE` page source, each displayed button applies to a specific **Product** element. What we want to do for the text of each button is this:

- If the product has not yet been bought, we want to display the text **Buy**. We can determine whether a product has been bought by checking if it exists as a **Purchase** element in the `$MT_IN-APP_PURCHASE` page source. This can be ascertained by checking the **SKU_ID** child element of **Purchase** elements to see if any matches the **SKU_ID** attribute of the current **Product** element.
- If the product has been bought, we want to distinguish between (i) subscription products on one hand and (ii) non-subscription products on the other. For subscription products that have been bought, we want to display the date till which the subscription is valid. For non-subscription products that have been bought, we want to display the purchase date.
- Subscription products will have an `@SubscriptionPeriod` attribute, so we can look for its existence to find out whether the product is a subscription product. Further, if the subscription is yearly, then the
value of the attribute will be $P1Y$ (that is, @SubscriptionPeriod="P1Y"); if it is monthly, then $P1M$ (that is, @SubscriptionPeriod="P1M"). Based on the value of the @SubscriptionPeriod attribute, we can calculate the expiry date by adding the appropriate period (year or month) to the purchase date.

- If a non-subscription product (consumables and non-consumables) has been bought, we want to display the purchase date.

The XPath expression for the Text property of the Buy button will therefore be that shown in the screenshot below. Note that the context node of the XPath expression will be the $SMT_IN_APP_PURCHASE/Root/Products/Product$ element. This is because the button is placed inside a dynamic table where rows repeat on the Product element.

```xml
1 let $sku := @SKU_ID
2 return
3 if ($SMT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID = $sku] and
4   (@SubscriptionPeriod="P1Y" or @SubscriptionPeriod="P1M"))
5   then concat("Valid to ", substring-before(xs:string(
6     $SMT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID = $sku]/@PurchaseTime)+ (if
7     (@SubscriptionPeriod="P1Y") then xs:yearMonthDuration("P1Y") else xs:yearMonthDuration("P1M"))), "T")
8 else (if ($SMT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID = $sku])
9   then concat("Bought on ", substring-before(xs:string(
10   $SMT_IN_APP_PURCHASE/Root/Purchases/Purchase[SKU_ID = $sku]/@PurchaseTime)), "T")
11   else "Buy")
```

**Purchase request to app store**

Depending on the text that is displayed on the Buy button, there are two possible ways to react to a button-click event:

- If the product has already been bought, then do not proceed with the in-app purchase
- If the product has not yet been bought, then go ahead with the purchase request to the app store.

The actions that have been defined for the button's on-click event implement this conditional processing and are shown in the screenshot below.
If the product has already been bought, a message to this effect is displayed. Otherwise, the Purchase action is executed, and the app sends a purchase request for the current product to the app store.

Note: The above scenario is a simplified one in that a product can only be bought once. Alternatively, you might want (i) to allow a product to be bought multiple times, or (ii) to check whether a subscription has expired or not before enabling/disabling a renewal.

22.7.6 The OnPurchaseUpdated Event

After an app store accepts a purchase request that has been made via the Purchase action, it sends information about the accepted purchase to the app. This information is stored in a Purchase element of the $MT_IN-APP_PURCHASE page source. After this information has been received from the app store, you would most likely need to carry out certain actions. It is to enable the possibility of triggering actions automatically upon receiving such "purchase-accepted" information from the app store, that the OnPurchaseUpdated event has been created. To set up actions that are triggered by this event, go to the project property In-App Purchase Actions, which opens the Actions dialog for the OnPurchaseUpdated event.

In our example, we have defined two actions to be carried out automatically upon receiving a "purchase-accepted" response from the app store:

- Re-count the number of purchases and update the count that is displayed in the app.
- Send an app-side acknowledgment of the in-app purchase to the app store. An acknowledgment is commonly required by app stores to complete the transaction and forward to the product vendor the money that was paid for the product. In the design, you can acknowledge an in-app purchase by specifying an Acknowledge Purchase action.
The screenshot above shows the OnPurchaseUpdated actions that have been defined in our example project. The acknowledgments have been made conditional on the specific product because (i) consumables and (ii) subscriptions/non-consumables require different types of acknowledgment.

**Note:** You might need to modify the actions above to suit different app store requirements.

### 22.7.7 Query/Restore Purchases

The Query Purchases and Restore Purchases (iOS) buttons retrieve information about the purchases of the current user from the app store. The buttons work by executing, respectively, the **Query Purchases** action and **Restore Purchases** action. The difference between the two actions is that the **Restore Purchases** action, in addition to retrieving in-app purchases carried out from the app on the current device, also restores purchases made by the current user on other devices. This is a requirement of the iOS app store and has therefore entailed that a separate action be created for the purpose.
In our simulation, the two buttons both update the Purchases element of the $MT_IN-APP_PURCHASE page source with information relating to purchases that is fetched from the simulation data file.
Additionally, note the following:

- The count of purchases is updated after the Query Purchases action or Restore Purchases action is carried out, and the count is automatically updated in the display.
Since the Restore Purchases button should be available only on iOS devices, the row containing the button has its visibility made conditional on the device being an iOS device. This has been done by setting the row's Visible property to check the value of the static variable $MT_iOS.
23 MT Solutions in UWP Apps

You can integrate MobileTogether solutions in Universal Windows Platform (UWP) apps. Each MobileTogether solution is placed in a UWP app via a SolutionView control, which Altova has introduced specially for this purpose. For example, the screenshot below shows the window of a UWP app which contains two SolutionView controls (outlined in red), each of which displays a MobileTogether solution. Each UWP app can integrate more than one solution (as in the screenshot below), all of which can run at the same time. Additionally, a UWP app can start a solution in a new UWP window.

The broad procedure for integrating MT solutions in your UWP app is as follows:
1. **Reference the MT Libraries**: Copy the relevant MobileTogether libraries to your project’s directory, and reference the MobileTogether libraries in your project.

2. Add one or more **SolutionView controls** to your XAML page or control.

3. **Create the code** to manipulate the SolutionView control programatically and to listen for events.

4. To interact with a MT solution hosted by SolutionView control, you can use MobileTogether's EmbeddedMessage mechanism in your code. See [Messaging in the UWP App](#) for information about how to send and receive messages from the MT solution.

5. Compile the UWP app.

This section is organized along the lines of the procedure given above.

For an example, see the topic [Example UWP App](#).
23.1 Reference the MT Libraries

In order to integrate MobileTogether solutions in your UWP app, you will need to reference the relevant MobileTogether libraries from your UWP project directory. These libraries will be located in a descendant folder of your MobileTogether application folder, so MobileTogether Designer will need to have been installed. Create the library reference as follows:

1. Inside the application folder, go to the subfolder MobileTogetherSPL\WindowsApp. The required libraries are in the ZIP file CustomAppTemplateWindows10.zip.
2. Copy the contents of the ZIP folder to your UWP project directory. The required files and folders are:
   - MobileTogether (folder)
   - zxing.uwp.ARM (folder)
   - zxing.uwp.x64 (folder)
   - zxing.uwp.x86 (folder)
   - MobileTogether.winmd
   - MobileTogether.pri
   - MobileTogether.Windows.ARM.dll
   - MobileTogether.Windows.x86.dll
   - MobileTogether.Windows.x64.dll
3. Add a reference to the MT libraries listed above, the best way being to manually edit the .vcproj file as listed below. Note that each implemented platform requires a different DLL.

```
<Reference Include="MobileTogether">
    <HintPath>MobileTogether.winmd</HintPath>
    <Implementation Condition=" '$(Platform)' == 'ARM' ">
        MobileTogether.Windows.ARM.dll
    </Implementation>
    <Implementation Condition=" '$(Platform)' == 'Win32' ">
        MobileTogether.Windows.x86.dll
    </Implementation>
    <Implementation Condition=" '$(Platform)' == 'x64' ">
        MobileTogether.Windows.x64.dll
    </Implementation>
</Reference>
```

For an example, see the topic Example UWP App.
23.2 XAML File

To add a SolutionView control to your XAML file, do the following:

1. Declare the MobileTogether namespace in your XAML page or control like this:

   ```xaml
   <Page
       xmlns:mobiletogether="using:MobileTogether">
   
   2. Add the SolutionView control:

   ```xaml
   <mobiletogether:SolutionView
       IsAutoSuspendResumeEnabled="False"
       IsBackKeyEnabled="False"
       IsEscapeKeyEnabled="False"
       IsEnterKeyEnabled="False"
       ServerURL="demo.mobiletogether.com"
       Port="443"
       UseSSL="True"
       User=""
       Password=""
       SolutionURL="/public/MyCollections?Par1=123&Par2=456"
       EmbeddedMessage="MyEmbeddedMessage"
       SolutionFinished="OnSolutionFinished" />
   
   For an example, see the topic [Example UWP App](#).

Properties of the SolutionView control

The control's properties are listed below:

- **IsBackKeyEnabled, IsEscapeKeyEnabled, IsEnterKeyEnabled**
  Since multiple SolutionView controls can run in a single UWP app window, all running solutions would process the same Back, Esc, and Enter keys. To ensure that these keystrokes are not processed for the wrong solution, these three attributes enable these keys to be disabled for any given SolutionView control.

- **ServerURL, Port, UseSSL**
  The URL and port of the MobileTogether Server on which the solution has been deployed. The UseSSL attribute specifies whether the server connection uses SSL or not.

- **User, Password**
  Specifies the credentials of the user that is accessing the solution. To specify that access is anonymous, use the empty string as the value of both attributes.

- **SolutionURL**
  Specifies the path of the solution on MobileTogether Server. In the listing above, notice how SolutionURL can contain a solution's input parameters.
**EmbeddedMessage**
The embedded message that is sent from the SolutionView control to MobileTogether Server. The data must be sent as a serialized JSON object. (If your data is in XML format, then the XML document must be wrapped inside a JSON object.)

**SolutionFinished**
When the solution is finished or closed, the `SolutionFinished` event is fired. You can use this event, for example, to close the UWP app window after the solution finishes.
23.3 Run MT Solution from Code

To run a solution, you can specify the required properties of the SolutionView control in the XAML file itself. Alternatively, you can run a solution from code (typically C++ or C#) as follows.

You can instantiate a SolutionView control in the XAML file with a name, as follows:

```xml
<mobiletogether:SolutionView Name="x:MySolutionView" />
```

You can then run the solution from code as follows:

```csharp
mySolutionView.ServerURL = "demo.mobiletogether.com";
mySolutionView.Port = "443";
mySolutionView.UseSSL = true;
mySolutionView.User = "";
mySolutionView.Password = "";
mySolutionView.SolutionURL = "/public/MyCollections?Par1=123&Par2=456";
await mySolutionView.StartSolution();
```

In code, the following actions can be performed via the corresponding methods listed below:

- Page submit: `mySolutionView.Submit();`
- Page refresh: `mySolutionView.Refresh();`
- Go back: `mySolutionView.GoBack();`
- Stop the solution: `mySolutionView.StopSolution();`
- Suspend the solution: `mySolutionView.SuspendSolution();`
- Access the solution's local storage: `mySolutionView.GetSolutionLocalFolder();`

Note the following points:

- The `SolutionURL` method can take the solution's input parameters.
- When the solution is finished or closed, the `SolutionFinished` event is fired. You can use this event, for example, to close the UWP app window after the solution finishes.
- Communication from the solution is carried out by writing to files in the solution's local storage or to any other accessible folder such as the Music, Videos, and Pictures folders.
- While it is possible to run the same solution in two separate SolutionView controls on the same page, there would be difficulties accessing the same resources. For example, you would encounter problems if you try to access a file in one SolutionView control while trying to delete it in another.

For an example, see the topic `Example UWP App`.
23.4 Messaging in the UWP App

The broad stages for messaging would be as follows:

1. Data is communicated from the UWP app to the solution (displayed in a SolutionView control of the UWP page).
2. The solution sends the data to the solution's workflow on MobileTogether Server, where it is processed in the usual way.
3. Results are returned to the SolutionView control, where they can be (i) displayed as part of the solution, and/or (ii) passed back to the UWP app for display or further processing.

Messaging mechanism

A message that is sent from a UWP app to the server is called an embedded message. The messaging mechanism uses three MobileTogether features:

- The OnEmbeddedMessage\(^{[367]}\) event, which is available as a page event;
- The $MT_EMBEDDEDMESSAGE JSON page source, which is generated automatically if at least one action is defined for the OnEmbeddedMessage\(^{[367]}\) event;
- The Embedded Message Back\(^{[758]}\) action, which sends a serialized JSON string as a return message;

How these features are used for messaging in the UWP app is as follows:

- SolutionView control now has a method ProcessEmbeddedMessage\(^{[337]}\) (msg) that (i) populates the $MT_EMBEDDEDMESSAGE JSON page source, and (ii) triggers the page event OnEmbeddedMessage\(^{[367]}\).
- SolutionView control now has an event named EmbeddedMessage\(^{[758]}\) that is triggered when, in the hosted solution, the Embedded Message Back\(^{[758]}\) action is executed. This event can be processed synchronously or asynchronously, for example, by using the standard UWP deferral mechanism as listed below.

```csharp
void EmbeddedMessage(object sender, EmbeddedMessageEventArgs e)
{
    var deferal = e.GetDeferral();
    ... Do some parallel processing of e.Message ...
    deferal.Complete();
}
```

Messaging: step-by-step

The steps given below provide a detailed look at the messaging process.

**UWP app to solution to MT Server**

1. The ProcessEmbeddedMessage\(^{[337]}\) method in your code\(^{[337]}\) can be used to transfer data from the UWP app to the SolutionView control. The data must be sent as a serialized JSON object. (If your data is in XML format, then the XML document must be wrapped inside a JSON object.)
2. The targeted solution must have an OnEmbeddedMessage\(^{[367]}\) event that has at least one action defined for it. In this case, when an embedded message reaches the solution, a $MT_EMBEDDEDMESSAGE JSON page source is created automatically and populated. The page source will have a root element named json, and its structure and content will come from the embedded message.
3. On the server, the $MT_{EMBEDDEDMESSAGE}$ JSON page source can be processed via actions of the `OnEmbeddedMessage` event or in any other way. This completes the first half of the transfer, from UWP app to MobileTogether Server. Processed data can now be sent back to the solution.

**MT Server to solution to UWP app**

4. The Embedded Message Back action sends a return message to the SolutionView control in the form of a serialized JSON string. In defining the action, you can specify the message that you want to send. For example, in the screenshot below, the action specifies that the message to be sent back is the entire data structure stored in the $MT_{EMBEDDEDMESSAGE}$ JSON page source: that is, the tree's `json` node and its contents.

```
Embedded Message Back
Message: $MT_{EMBEDDEDMESSAGE}$
```

5. When the solution's Embedded Message Back action is executed, the SolutionView control's `EmbeddedMessage` event is triggered. The embedded message received by the SolutionView control can now be processed by the code of your UWP app.
23.5 Example UWP App

Your MobileTogether Designer installation contains an example C# project that shows how to use the SolutionView control in a simple UWP app. This project is available in the (My) Documents folder: Altova\MobileTogetherDesigner7\MobileTogetherDesignerExamples\WindowsClientControl.

To run this sample UWP app, do the following:

1. Go to the C:\Program Files (x86)\Altova\MobileTogetherDesigner7\MobileTogetherSPL\WindowsApp folder located in the MobileTogether Designer application folder and copy the MobileTogether libraries in this folder.
2. Go to the folder where the C# project is located and unzip the MobileTogether libraries there.
4. Load the C# project. Compile it and run it.
24 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.
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.
24.1 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.

Note: You can localize services for other languages via the Localization dialog (Project | Localization menu command).

The $MT_SERVICE page source
The $MT_SERVICE page source is automatically created when the service design is created. The screenshot and listing below show the structure of the page source.

![Structure of $MT SERVICES page source](image)

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.
24.2 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.
24.3 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).

Service settings for /services/MTSLogs

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.
25 Menu Commands

MobileTogether Designer menu commands are organized into the following menus:

- File
- Edit
- Project
- Debug
- Page
- Table
- View
- Tools
- Window
- Help
25.1 **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**
- **Export MobileTogether Package**
- **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**

### 25.1.1 **New**

*Icon*  

*Shortcut*  

Ctrl+N

*Description*

Opens a new document tab in the main window and loads an empty MobileTogether Design file into this tab. The document is stored temporarily in memory. It must be saved to disk with the `.mtd` extension if you want to keep it.
25.1.2 New Service

- **Icon**

- **Description**
  
  Opens a new document tab in the main window and loads an empty MobileTogether Server Service file into this tab. The document is stored temporarily in memory. It must be saved to disk with the `.mtd` extension if you want to keep it. For a description of the MobileTogether Server Service features, see the section [Server Services](#).

25.1.3 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:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Icon] Checked in. Available for check-out.</td>
<td></td>
</tr>
<tr>
<td>![Icon] Checked out by another user. Not available for check-out.</td>
<td></td>
</tr>
<tr>
<td>![Icon] Checked out locally. Can be edited and checked-in.</td>
<td></td>
</tr>
</tbody>
</table>

- 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.

### 25.1.4 Reload

**Icon**

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.

### 25.1.5 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.

### 25.1.6 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.

![Open dialog](image)

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.

- 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.

25.1.7 Export MobileTogether Package

Opens the Export Package dialog (screenshot below) for creating a MobileTogether package file (.mtp file) from the active MobileTogether design. The MobileTogether package is a zipped archive that contains the design file in an encrypted form and, optionally, additional deployable resource files that are used by the design (such as images and CSS stylesheets). See MobileTogether Packages for more information.

In the Export Package dialog (screenshot above), select the resources you wish to include in the MobileTogether package:

- **Deployable files**: These resources will be exported with the design. Note that deployable files are saved in MobileTogether's database on the server and cannot be modified. Typically these are image files or files containing unchanging data. Files that cannot be deployed would be data files that are modified...
during a solution run, and are, therefore, not stored in the MobileTogether database on the server. Such non-deployable files must be saved separately to the server. See Deploying the Project for more information.

- **Workflow Icon file:** This option is enabled only if a Workflow Icon file has been set in the project's properties.

- **Global resources with active configuration:** Select the active configuration you want to use for global resources. The corresponding global resources will be added to the package.

**Note:** Test cases that have been saved with the design are not exported to the package.

Click Export when ready. In the dialog that then appears, enter the location where you want to save the package file and click Save. The design and the selected resources will be saved in a MobileTogether package file that has a .mtp file extension. See MobileTogether Packages for information about using package files.

### 25.1.8 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.

- **Design language:** Select the language that will be used by default when the solution is opened on a client. The dropdown list of the combo box displays the languages that have been defined in the design's Localization dialog. *Auto* refers to the design's default language, which is the language in which the solution is designed. The other items in the dropdown list are the names of the localizations (the languages into which the design's default language is translated). Note that, after deploying the solution to MobileTogether Server, you can change the default starting language of the solution in the Workflows tab of MobileTogether Server.

- **Active configuration of global resources:** Select the active configuration from the available configurations in the dropdown list of the combo box. The available configurations are those that are defined in the Global Resources Definitions file and are automatically obtained from there. See the section Altova Global Resources 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.

### 25.1.9 Open from MobileTogether Server

**Icon**

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.

### 25.1.10 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**.
25.1.11 Generate Program Code for AppStore Apps

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.
Menu Commands

General

Build Modes

- Publish
  The packages generated by this build mode are intended for final release. The solution will be deployed on the server, and the packages can be made available in App Stores for customers.

- Trial Run on Client
  The packages generated by this build mode can only be used for testing with a Designer as server. The solution will not be deployed on the server but will be loaded directly from the Designer. These packages should never be uploaded to App Stores for customers.

App

The name of the executable file, which must consider the different platform constraints (e.g. "MyProductApp")

Executable file name: MyFirstApp

The name visible on the Home screen on the client (e.g. "My Product App")

Visible name: My First App

App version number, must be integral due to AppStore limitations

Version: 1

App languages (only Windows App, Windows Phone)

The languages that the app supports in addition to English.

The app’s user interface, including error messages, will appear in these languages only. The Windows App/Phone Store will require you to provide a description in each chosen language.

This setting is independent of the solution localization defined in the Localization dialog – all languages you provide there remain available.

Supported languages: ✔ French ✔ Japanese
✔ German ✔ Spanish

For starting App from URLs (optional)

The URL scheme for starting your app via a link, e.g. mobiletogogether in mobiletogogether://mt/run-solution, generated by XPath function mt-run-appstoreapp-url

URL scheme: myappscheme

The URL host for starting your app via a link, e.g. mt in mobiletogogether://mt/run-solution, generated by XPath function mt-run-appstoreapp-url

URL host: myfirstapp

< Back Next > Finish... Cancel
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**).

### 25.1.12 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.

### 25.1.13 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*).

### 25.1.14 Print Preview, Print Setup

**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. 

© 2015-2021 Altova GmbH  
Altova MobileTogether Designer
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.
25.1.15 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.
25.2 Edit

The Edit menu contains the following commands:

- Undo
- Redo
- Cut
- Copy
- Paste
- Delete
- Select All

25.2.1 Undo, Redo

- **Undo**
  - **Icon**
  - **Shortcut** Ctrl+Z
  - **Description** Supports unlimited levels of Undo. Every action can be undone and it is possible to undo one command after another. The Undo history is retained after using the Save command, enabling you go back to the state the document was in before you saved your changes. You can step backwards and forwards through this history using the Undo and Redo commands (see Redo command below).

- **Redo**
  - **Icon**
  - **Shortcut** Ctrl+Y
  - **Description**
Allows you to redo previously undone commands, thereby giving you a complete history of work completed. You can step backwards and forwards through this history using the **Undo** and **Redo** commands.

### 25.2.2 Cut, Copy, Paste, Delete

#### Cut
- **Icon**
- **Shortcut**
  - Ctrl+X or Shift+Del
- **Description**
  - Copies the selected text or items to the clipboard, deleting the selection from its current location.

#### Copy
- **Icon**
- **Shortcut**
  - Ctrl+C
- **Description**
  - Copies the selected text or items to the clipboard, *without* deleting the selection from its current location. The command can be used to duplicate data within MobileTogether Designer, or to move data to another application.

#### Paste
- **Icon**
25.2.3 Select All

- **Shortcut**
  - Ctrl+A

- **Description**
  - Selects the contents of the entire document.
25.3  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
- List Page Sources by Attribute
- 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
- Replace DB Sources
- Cache Overview
- Localization
- Simulation Language
- List All File and Directory References
- List All External Data References
- List Unused Functions, User Variables, Etc
- Maintain OAuth Settings
- Import OAuth Settings
- iOS Push Notification Button Sets
- In-App Purchase Products

25.3.1  Validate

**Icon**

**Description**

Validates the currently active project. If a project contains multiple pages, all the pages are validated. Validation results are reported in the *Messages Pane* in terms of the number of errors and warnings. If an error or warning is detected, a message about each is displayed.
25.3.2  Reload Page Source Structures

Reloads the structures of all page sources of the project. This command is useful if you want to reload all pages sources at once. Additional options for reloading page source structures are available in the File tab of the Options dialog.

25.3.3  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.
25.3.4  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.

25.3.5 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.

![Server Settings dialog](image)

### 25.3.6 Record New Test Case

**Icon**

![Icon](image)

**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.

### 25.3.7 Playback Test Case

**Icon**

![Icon](image)

**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.

### 25.3.8 Trial Run Test Cases on Client

**Icon**

![Icon](image)

**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.

### 25.3.9 Manage Test Cases and Runs

**Icon**

![Icon](image)

**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.

25.3.10 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. 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.

- 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.
25.3.11  List Usages of All Global Variables

Returns a list, in the Listings 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.

25.3.12  List Usages of All Page Source Variables

Returns a list, in the Listings 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.

25.3.13  List Page Sources by Attributes

Returns a list, in the Listings Pane, of the current project's page sources, organized by page source attributes (that is, according to the values of the project's various settings). Two lists are returned.

List by single attribute (value)

If an attribute value has been set for any of the project's page sources, the attribute is listed together with the page sources that have this value set. The following page source attributes are evaluated:

- Client only, Server only, or Shared
- Read-only or Editable (Mutable)
- Persist on client or not
- Load behavior
- Save behavior
- Reset Behavior
List by coincidence of values of all attributes
The page sources are grouped together if they have the same values for all attributes. See screenshot below.

Clicking a page source link highlights the page source in the Page Sources Pane.

25.3.14 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 \textbf{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.

\textbf{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 \textit{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.

\section{25.3.15 List Usages of All User-Defined XPath/XQuery Functions}

Returns a list, in the \textit{Listings 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.

\section{25.3.16 Action Groups}

- \textit{Icon}

- \textit{Description}
Displays the Action Groups dialog (*screenshot below*), in which action groups can be created and edited.

To create an Action Group, do the following:

1. Click **Manage** in the Action Groups pane (*see screenshot below*).

2. In the Manage Group Actions dialog that appears, click **Add** in the toolbar of the dialog. An Action Group with a default name is added to the list of Action Groups in this dialog.

3. Double-click the name of Action Group to a suitable name to change it, and click **OK**. The Manage Action Groups dialog closes, and the new Action Group is added to the list of groups in the Action Groups pane (*see screenshot above*).
4. In the Action Groups pane (*screenshot above*), click the **Edit** icon of the Action Group you want to edit. The group's contents are displayed in the pane on the right (*see screenshot below*). The details of an Action Group can also be displayed by selecting the group in the Action Group combo box (*see screenshot below*).

![Action Group Editor](image)

5. To edit the contents of the active Action Group in the right-hand pane, drag and drop actions 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.

### 25.3.17 List Usages of All Action Groups

Returns a list, in the **Listings 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.
25.3.18 Style Sheets

Displays the Style Sheets dialog (screenshot below), in which you can manage style sheets and define the styles in the different style sheets.

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 want an overview of currently defined styles. The left-hand pane also has toolbar icons for (i) expanding all items, and (ii) collapsing all items.

For an overview of how style sheets work, see the section *Style Sheets*.

### 25.3.19 List Usages of All Style Sheets

A **user-created style sheet** can be applied to page instances, table instances, and control instances. This command returns a list, in the Listings 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.

### 25.3.20 Rich Text Style Sheets

Opens the Rich Text Style Sheets dialog (screenshot below), in which you can define:

- Styling rules for text in the solution display
- Mappings of styles from the page source to the solution display, and vice versa
For a description of the Rich Text Style Sheets dialog, see the Rich Text section.

25.3.21 Replace DB Sources

The **Replace DB Sources** command enables you to switch the database connection of a DB page source. This can be useful, for example, if you wish to switch from a test environment to a production environment. In order for the DB switch to be successful, the two DBs must be identical in type and structure, and the following conditions must be met:

- The type of the original and replacement DBs must be the same. For example, both DBs would be MS SQL Server.
- The name and structure of the root object of both DBs must be the same.
- The structure of the selected objects in both DBs must be the same.

**Note:** If the root object is not the same or if the structures of objects do not match, then no error or warning is reported. We therefore recommend that you check whether the switch has been successfully implemented.

**How to replace a DB connection**

When the command is executed, the Replace Database Connections dialog appears (see screenshot below). The dialog lists the DB connections that are currently active in the design, with each connection showing the number of page sources and the number of actions that use that DB connection.

To replace one of the listed DB connections, select it and click **Replace**. This opens the Database Connection Wizard, with the help of which you can define the replacement connection that you want.
25.3.22 Cache Overview

- **Icon**

- **Description**

  Connects to MobileTogether Server and provides an overview of server caches in the Cache Overview dialog (screenshot below).

The dialog provides, in the top pane, an overview of all the caches on the server. In the dialog, you can do the following:
Menu Commands

Project 1423

- Activate/deactivate a cache (in the top pane).
- Delete a cache (in the top pane) by selecting it and clicking **Delete**.
- Select a cache in the top pane to display its details in the other (subsidiary) panes of the dialog.
- Delete a cache item in a subsidiary pane by selecting it and clicking the subsidiary pane's **Delete** button.
- Reconfigure a cache by clicking the **Additional Dialog** button (at the right-hand side of the cache entry in the top pane), and editing the cache's configuration.
- View a log of cache entry updates by clicking the **Additional Dialog** button in the Cache Entries pane (located at the bottom right of the dialog). In the screenshot above the MyCars cache has three cache entries, which are listed in the Cache Entries pane.

See the section Caching for more information about caching.

### 25.3.23 Localization

#### General description

Displays the Localization dialog (screenshot below), in which you can localize (translate) strings. These would be strings that either appear in various controls (for example, the text of a button) or are related to controls in other ways (for example, the dropdown list values of combo boxes). In addition to strings related to controls, any custom string can be localized and subsequently inserted at any location in the design via the **mt-load-string** function.

- **Control-related strings** are displayed in the upper pane (see screenshot below). The columns of this pane are, respectively, from left to right: (i) the page in which the control appears; (ii) the control's name; (iii) the control property that contains the text string; (iv) the text string in the default language; additional columns contain the text string in the localized languages, one column per language.
- **Custom strings** are displayed in the lower pane. The columns of this pane are, respectively, from left to right: (i) the custom string's name; (ii) the custom string in the default language; additional columns contain the custom string in the localized languages, one column per language.

When the dialog opens, all strings are listed by control, and the first string of the control that had the focus (when the dialog was opened) will be selected in the dialog.

You can add a column for a localization language with the **Add Language** icon. In the dialog that appears, enter the language code (used for internal) and language name (used in MobileTogether user interfaces). You can add as many columns as you like. To localize a control-related string or custom string in a particular language, enter the translation in the column for that language. To save translations, click **OK**.

All strings (control-related and custom) that have been localized in the Localization dialog will be used in the localized versions of the solution. If the language setting on the mobile device specifies a **language-country** variant (for example, **es-US** or **fr-CH**), then the solution's language is selected according to the cascading order given below:

1. If the solution contains a matching **language-country** (**es-US** or **fr-CH**) localization, then strings of this localization are used where these exist.
2. If no matching language-country localization (es-US or fr-CH) exists for a string, then the localized language (es or fr) string is used—if one exists.

3. If no language-country localization (es-US or fr-CH) or language localization (es or fr) exists for a string, then the default language of the solution is used for that string.

If you wish to see a simulation in any of the languages for which localized strings are defined, set the simulation language via the Project | Simulation Language command, and then run a simulation.

Apply Filter

The following filters are available in the Localization dialog; they can be combined:
Menu Commands

- **Page**: Select a page from the dropdown list to see the strings of only that page. To select all pages, leave the option blank or select the empty entry. This filter applies to control-related strings (upper pane) only.
- **Control-related strings and custom strings**: Enter the text to search for. All control names (second column of upper pane) and custom string names (first column of lower pane) that contain the search text are displayed. This filter applies to both panes.
- **Incomplete translations only**: Check this option to show only incomplete translations. An incomplete translation is a string with at least one localization missing. This filter applies to both panes. Note that, if a translation is entered while this option is selected, then the display must be refreshed in order to update the filtering. You can refresh the display by deselecting and then re-selecting the option.
- **Search in translations**: If this check box is selected, the search (for the term entered) is carried out in both the default language and translations. The search is carried out while typing the search term.

**Adding, renaming, and deleting language columns**

These icons are located above the upper pane, on the right-hand side.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Add Language</th>
<th>Rename Language</th>
<th>Delete Language</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add Language" /></td>
<td>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. <em>(See also the related mt-available-languages XPath function.)</em></td>
<td>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.</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Add Localization String</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add Localization String" /></td>
<td>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. <em>(See below.)</em></td>
</tr>
</tbody>
</table>
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**

```xml
<Localizations version="1">
  <Controls>
    <Control default="Select Employee" property="Page Title" name="Select Employee" id="2">
      <Languages de="Mitarbeiter auswählen"/>
    </Control>
    ...
  </Controls>
  <Strings>
    <String default="Admin" name="Role-A Name">
      <Languages de="Admin"/>
    </String>
    ...
  </Strings>
</Localizations>
```

**XML listing of multiple-language export**

```xml
<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 the existing strings are non-empty.
- If you select the Import default text translations option, then incoming translations that are the same as the default-language text will be imported (as with the Admin string for German (de) in the listing above). Otherwise, nothing will be imported for such fields, and the existing text in these fields will not be modified. (The Import default text translations option is enabled only if the option to export the strings of the default language was selected during the Export process; see description of Export above.)
- Click Import to finish.

The translated language strings are imported into the project and entered into the corresponding localization-language column/s of the Localization dialog. Note that the structure and content of the imported XML file must be such that MobileTogether Designer can correctly process the XML file. It is therefore important not to change the values of any other attributes besides the localization-language attributes.

Note: In the XML file, if a translated string is identical to the default-language string, then the translated string is not imported and the corresponding entry in the localization-language column is
empty. When the solution is run, since there is no localization-language entry, the default-language string will be used. In this way, duplications of strings across languages are avoided.

25.3.24 Simulation Language

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.

25.3.25 List All File and Directory References

Returns a list, in the Listings 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.

25.3.26 List All External Data References

Returns a list, in the Listings 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.

25.3.27 List Unused Functions, Variables, Etc

Returns a list, in the Listings Pane, of all the unused XPath/XQuery functions, user variables, page sources, pages, 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.
25.3.28 Maintain OAuth Settings

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 a MobileTogether Designer project (see the documentation here). These are stored in a pool, and you can use a definition from the pool for authenticating REST requests defined anywhere in the document. The Maintain OAuth Settings dialog (screenshot below) lets you manage the OAuth definitions of the active project. The dialog displays all the OAuth settings definitions that are currently in the active project's pool of definitions. You can delete definitions from the pool, import definitions from other open MobileTogether Designer projects, copy definitions to the clipboard, and paste definitions from the clipboard.

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.
Menu Commands

- **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.

### 25.3.29 Import OAuth Settings

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 (see screenshot below) enables you to import definitions from other open MobileTogether Designer projects into the current project.

![Import OAuth Settings](image)

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.
25.3.30  iOS Push Notification Button Sets

This command is used to create the buttons that appear in push notifications (PNs) received on iOS devices. iOS PN buttons are available in AppStore Apps, 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).

![iOS Push Notification Button Sets dialog](image)

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.

---

25.3.31  In-App Purchase Products

This command opens the In-App Purchase Products dialog (screenshot below). Here you can map a product name (in the Product column) to the SKU IDs of the product in the respective app stores (Google, Apple, Windows). (These IDs are assigned when you create/register the product in your app store account.) In the MT
design, each product is identified by the name you give it in this dialog. When the respective AppStore Apps are generated from the design, the appropriate SKU ID is used for the app on each platform.

To add a new mapping, click the **Plus** icon. To remove a mapping, click the **Delete** icon.

See the topic Register Products for details.
25.4 Debug

The **Debug** menu contains commands that apply to MT Debugger and the debugging of XPath expressions and actions. It contains the following commands:

- **Resume Debugging / Go**
- **Stop Debugging**
- **Step into Action**
- **Step into XPath**
- **Step Out**
- **Step Over**
- **Run to Selected Action**
- **Stop at Next Error**
- **Stop at Next Breakpoint**
- **Stop at Next Action**
- **Disable All Breakpoints**
- **Enable All Breakpoints**
- **Remove All Breakpoints**
- **Debug Windows**

### 25.4.1 Resume Debugging / Go

- **Icon**

- **Shortcut**
  - F5

- **Description**
  
  Starts the **Actions Debugger** or **XPath Debugger**, whichever is active. It can also be used to resume debugging where this has been paused. The command is enabled only when one of these debuggers has been opened from a simulation. See **MT Debugger** for more information.

### 25.4.2 Stop Debugging

- **Icon**

- **Shortcut**
Shift+F5

- **Description**
  Stops debugging and closes MT Debugger. It is enabled when MT Debugger has been started, that is, if (i) a debugger mode has been selected, and (ii) a simulation has been started.

### 25.4.3 Step into Action

- **Icon**

  ![Icon](image)

- **Shortcut**
  F11

- **Description**
  Proceeds through the action execution in Actions Debugger, one step at a time. The command is enabled after the Actions Debugger has been opened.

### 25.4.4 Step into XPath

- **Icon**

  ![Icon](image)

- **Shortcut**
  Ctrl+Shift+F11

- **Description**
  Opens the XPath Debugger and displays the XPath expression of the action. The command is enabled after the Actions Debugger has been opened.
25.4.5 Step Out

- Icon

- Shortcut
  Shift+F11

- Description
  Steps out of the current action execution step, and goes to the parent step. The command is enabled after the Actions Debugger has been opened.

25.4.6 Step Over

- Icon

- Shortcut
  Ctrl+F11

- Description
  Steps over descendant steps of the action execution. The command is enabled after the Actions Debugger has been opened.

25.4.7 Run to Selected Action

- Icon

- Shortcut
  Ctrl+F5

- Description
The command is enabled after the Actions Debugger has been opened. If the command is clicked, then the Debugger executes all actions up to the current cursor location and stops there. If a breakpoint is encountered at an earlier point, then the debugger stops at this point and can be resumed by running the Run to Selected command again. If the action at the current cursor location cannot reached during execution, then action execution proceeds without stopping.

25.4.8 Stop at Next Error

Icon

Description
Stops the simulation at the next XPath error and displays the XPath expression in XPath Debugger. This command enables you to specify the debugger mode of a simulation before you start it. You can also use this command to change the debugger mode after a simulation has been started.

25.4.9 Stop at Next Breakpoint

Icon

Description
Stops the simulation at the next breakpoint (which may be on an action or XPath expression) and opens the appropriate Debugger. This command enables you to specify the debugger mode of a simulation before you start it. You can also use this command to change the debugger mode after a simulation has been started.

25.4.10 Stop at Next Action

Icon
Description

Stops the simulation at the action/s of the next event that is triggered and displays the action/s in the Actions Debugger. This command enables you to specify the debugger mode of a simulation before you start it. You can also use this command to change the debugger mode after a simulation has been started.

25.4.11 Disable All Breakpoints

Breakpoints can be temporarily disabled, in which case they will be ignored during debugging. This command disables all breakpoints that have been set on actions and on XPath expressions. You can also disable all (or individual) breakpoints in the Breakpoints Pane. For more information about breakpoints, see MT Debugger.

25.4.12 Enable All Breakpoints

Breakpoints can be temporarily disabled, in which case they will be ignored during debugging. This command ensures that all breakpoints are enabled, including those that have been disabled. It applies to all breakpoints, whether set on actions or on XPath expressions. You can also enable all (or individual) breakpoints in the Breakpoints Pane. For more information about breakpoints, see MT Debugger.

25.4.13 Remove All Breakpoints

Remove all breakpoints in the design, whether set on actions or on XPath expressions. You can also remove all (or individual) breakpoints in the Breakpoints Pane. For more information about breakpoints, see MT Debugger.

25.4.14 Debug Windows

Placing the cursor over the Debug Windows causes a sub menu to appear, which has the following commands:

- **Breakpoints**: The Breakpoints Pane is, by default, tabbed together with the Controls Pane. Click this command to make the Breakpoints Pane the active tab.
- **Pin Debugger Action View**: This command is enabled only when MT Debugger has been started, that is, if (i) a debugger mode has been selected, and (ii) a simulation has been started. It makes the Actions Debugger the active window.
- **Pin Debugger XPath View**: This command is enabled only when MT Debugger has been started, that is, if (i) a debugger mode has been selected, and (ii) a simulation has been started. It starts the XPath Debugger and makes it the active window.
25.5 Page

The Page menu contains commands that apply to the currently active page of a project. You must click in the page to enable the commands of the menu. It contains the following commands:

- Page Actions
- Actions Overview
- Jump to Control
- Show/Define Tab Order
- List Text Size Auto-Fit Groups
25.5.1 Page Actions

Displays the Actions dialog of the currently active page (see screenshot below). The left-hand pane of the dialog contains the available actions, organized by functionality. The right-hand pane contains tabs of available events for that page. The events that are available depend on the role of that page in the project workflow. For instance, a page that cannot be returned to by pressing the Back button will not have the OnBackButtonClicked event tab.

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.
25.5.2 Actions Overview

Displays the Actions Overview dialog of the currently active page (see screenshot below). The dialog shows Control Actions and Page Actions, as well as actions you can set for the project. Each control in the page design is listed, together with its event/s and corresponding action/s. If an action has been defined for an event, then that action is listed in the Actions column; otherwise no action is listed for the event. The screenshot below shows that only one action has been defined on a combo box event.

The dialog is structured to show:

- Actions of project-wide events
- Page Actions. In the screenshot above, for example, SplashScreens is the name of the project's one page. All the events of the page are listed. No action has been defined for any of these events.
- Control Actions. Each control in the design is listed with its respective event.

This dialog not only provides you with an overview of all actions defined in the design, but also enables you to do the following:

- To create an action for any event or to edit an existing action: Click the Edit icon of that event (see screenshot above). This takes you to the corresponding Actions dialog of the control, page, or project.
- When a control is selected (not a page or the project), then the Jump to Control button is enabled. Clicking it takes you to that control in the design.
**Note:** The last selection in this dialog is remembered. As a result, the dialog is always reopened with the last selection highlighted.

### 25.5.3 Jump to Control

- **Shortcut**
  - Ctrl+J

- **Description**
  Displays the Jump to Control dialog (*screenshot below*). The dialog contains a combo box with a dropdown list that shows all the controls of the currently active page design. Page controls are listed alphabetically by the value of their Name properties.

Select a page control from the dropdown list or enter the control's name (auto-completion is available). Click **OK** to finish. The page control that was selected in the dialog box will now be selected in the design. If the control is associated with a data node, that node in the Page Sources Pane will also be selected.

### 25.5.4 Show/Define Tab Order

- **Shortcut**
  - Ctrl+T

- **Description**
  The Tab Order feature enables you to define a tab order for the controls on the page. Once defined, every time the user of the device (Web and Windows clients only) taps the **Tab** key, the focus of the solution jumps to the next control in the tab order sequence.

  When selected, this command does the following: (i) displays the Tab Order Options dialog (*screenshot below*), (ii) shows a blue number icon on all controls in the design that may be assigned a number in the
Menu Commands

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.

25.5.5 List Text Size Auto-Fit Groups

Displays in the Listings 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.
25.6 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
- Border Settings

For more information about tables, see the section Design Object/Features | Tables.

25.6.1 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.
The **Insert Table** command inserts a static table of dimensions 2x2 to the left of the selected cell.

The **Delete Table** command deletes the currently selected table.

### 25.6.2 Insert/Append/Delete Row/Column

The two **Insert**, two **Append**, and two **Delete** commands shown in the table below are available when a row or column of any kind of table (static, repeating, or tables with dynamic rows and/or dynamic columns) is selected.

<table>
<thead>
<tr>
<th>Insert Row</th>
<th>Insert Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append Row</td>
<td>Append Column</td>
</tr>
<tr>
<td>Delete Row</td>
<td>Delete Column</td>
</tr>
</tbody>
</table>

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.
25.6.3  Join/Split Cells

The four Join commands and two Split commands (shown in the table below) enable you to, respectively, join the currently selected cell with an adjacent cell or split the currently selected cell. They are available for the cells of all tables (static, repeating, or tables with dynamic rows and/or dynamic columns).

<table>
<thead>
<tr>
<th>Join</th>
<th>Split</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join Cell Left</td>
<td>Split Cell Horizontally</td>
</tr>
<tr>
<td>Join Cell Right</td>
<td>Split Cell Vertically</td>
</tr>
<tr>
<td>Join Cell Above</td>
<td></td>
</tr>
<tr>
<td>Join Cell Below</td>
<td></td>
</tr>
</tbody>
</table>

**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 cell will be empty.

25.6.4  Show Add–Remove Buttons

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 page source. You can add a new row by tapping the Add button.

<table>
<thead>
<tr>
<th>ID</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Vienna</td>
</tr>
<tr>
<td>21</td>
<td>Munich</td>
</tr>
<tr>
<td>22</td>
<td>London</td>
</tr>
<tr>
<td>23</td>
<td>Paris</td>
</tr>
<tr>
<td>24</td>
<td>Boston</td>
</tr>
<tr>
<td>25</td>
<td>Tokyo</td>
</tr>
<tr>
<td>28</td>
<td>Moscow</td>
</tr>
</tbody>
</table>

### 25.6.5 Add Header/Footer, Leading/Trailing Column

These four commands each add, respectively, a table's header, footer, leading column, or trailing column. They each apply to tables with either dynamic (repeating) rows 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.)*

### 25.6.6 Remove Header/Footer, Leading/Trailing Column

These four commands each remove, respectively, a table's header, footer, leading column, or trailing column. They each apply to tables with either dynamic (repeating) rows 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.)*
25.6.7 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.

25.6.8 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.

25.6.9 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.

### 25.6.10 Border Settings

The **Border Settings** command opens the Border Settings dialog (screenshot below), in which you can set border properties for the selected table item/s and related table items (for example, the table items that are related to a cell are the rows, columns, row groups, and table that contain the cell).
The mechanism to set border styles works as follows:

1. In the design, select the table item or multiple table items that you want to style: cell/s, row/s, column/s, row-group/s, table.
2. In the Line Settings pane, define the three border properties: width, color, and style that you want to apply.
3. The Border Level combo box of the Borders pane displays table items according to the table item you selected in Step 1. In the combo box, choose the selected table item or one of the options for related table items. The style will be applied to the table item/s that you select in the combo box.
4. **Apply the border styles** to one or more individual borders (top, right, bottom, left) of the currently selected table items. Use the Presets pane and/or Borders pane to apply styles. Note that the buttons in these two panes are toggles; they apply/remove styles to the respective border.

5. To set new border styles (for non-styled borders or already styled borders), repeat Steps 1 to 4.

6. Click **Close** to finish.

These steps are explained in more detail below.

**Selection of table item/s**

Select table items in the design as follows:

- **Cell:** Click inside the cell.
- **Multiple cells:** Click the first cell; keep the Ctrl key pressed and click the other cells that you want to make part of the selection.
- **Row, column, row-group:** Click inside a cell of the respective item. Alternatively, click the border of the item.
- **Multiple rows, columns, row-groups:** Select the first item; keep the Ctrl key pressed and click the other items.
- **Table:** Click inside a cell of the table. Alternatively, click the border of the table.

After the table item is selected in the design, go to the **Border Level combo box** and choose the context-relevant items to which the styles should apply.

**Line settings**

Borders have three properties: width, color, and style. You can select the value of each property from its combo box. Alternatively, you can enter an XPath expression to select the value of a property.

The resulting border style is displayed as a preview. In the screenshot above left, for example, the preview line is green, dashed, and has a width of 3px. If a property is specified as an XPath expression, then this is indicated by **Pooled XPath**, and, since the property value is not known till run time, the property is indeterminate in the preview. (An example of such a preview is the screenshot above right, where the style
property is dashed (and shown as such), whereas the width and color are indeterminate.) At run time, the evaluated values will be applied to the respective border properties.

**Border level**

In the Borders pane, the Border Level combo box enables you to select items that are related to the table items currently selected in the design *(see screenshot below)*.

The combo box items from which you can choose are contextually related to the table item/s currently selected in the design. To the right of each item in the combo box, additional information provides detail about the table structure. For example, we can see that, in the screenshot above, the items that are selected in the design are two adjacent cells in a row of Table2. The information in the last line, 2x1 items, indicates that the two cells are each in a separate column but in a single row. If the two cells were in one column and in two rows, the information would read 1x2 items. (The first number is the number of columns, the second is the number of rows.) Note that the Rows and Columns item also shows information in the column x row format.

In the combo box, select the table item/s to which you want to apply the style you have defined in the Line Settings pane. If, for example, you select Row, then the style will be applied to the border/s of the currently selected row. If you select Cells, then the style will be applied to the border/s of the currently selected cells, individually.

**Applying styles to (individual) borders of selected table items**

The style that is currently defined in the Line Settings pane can be applied in two ways to borders of the table items selected in the Border Level combo box:

- By selecting one of the presets in the Presets pane
- By selecting a combination of individual borders in the Borders pane

The options provided by the two panes are described below.

**Note:**  The application options in both panes are toggles. This means that, if the current style (exactly as defined in the Line Settings pane) has not been applied, then clicking an option applies the current style; if the current style has been applied (exactly as defined in the Line Settings pane), then clicking an option removes the current style.
Presets pane
Three preset options (see screenshot below) are available, and are described below. Note that only applicable options are enabled.

- **None**: This preset option is always enabled. It removes all border styles from the selected table item. If you want to remove border styles from any table item, this option is the best way to do it.

- **Outline**: This preset option is always enabled. It applies the current style to the outer border of the selected item (on all sides). If the selected item is a table, then the table border gets the style. If two cells are selected, then both cells, individually, get a border in the current style.

- **Inside**: This option is enabled only when at least two adjacent cells/rows/columns are selected; it is not applicable to tables. For example: (i) if applied to two columns, then the current style is applied only to the inside vertical borders of all cells in the column; (ii) if applied to two horizontally adjacent cells, then the inside vertical borders of both cells get the style; (iii) if applied to two vertically adjacent cells, then the inside horizontal borders of both cells get the style.

Borders pane
Options in the Borders pane enable you to apply the current style to individual borders of a table item (that is, top, right, bottom, left borders separately). Each individual border has its own icon. Click an icon to apply the current style to that border.

Note the following points:

- For each table item selected in the Border Level combo box, a relevant set of icons is enabled. For example, if a single cell is selected, then only four icons will be enabled (corresponding to the top, right, bottom, and left borders). If, on the other hand, however, All Rows and Columns is selected, then all 14 icons in the pane are enabled (see screenshot below). Note that the inner borders can be styled separately (for example: bottom or top inner row borders) or together (bottom and top inner row borders). Also note that, strictly speaking, columns themselves are not given a top or bottom border; it is the topmost cell in the column that is given a top border and the bottom-most cell of the column that is given a bottom border. This works analogously for the left and right borders of rows.
- Cells of a column or row will inherit the style of that column or row, respectively.
- Individual table borders can also be applied by moving the mouse, in the Borders pane, over the location of the border you want to style. When the border becomes highlighted in yellow (see screenshot above), click to apply the current style. If you move the mouse over the intersection of two borders, then both borders become highlighted (see the first screenshot at the top of this page), and you can apply the current style to both borders at once.
- After a style has been applied to a border and you move the mouse over the border, in addition to the border becoming highlighted in yellow, the border's properties are displayed in a popup (see screenshot above).
- When you click an icon or a highlighted border, the current style is applied if that border does not have the same style as the current style. If the border already has the current style and you click the border or its icon, then the current style is removed from that border.
25.7 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

25.7.1 Status Bar and Panes

The display of the Status Bar and various panes can be toggled on/off by clicking their commands in the View Menu:

- Status Bar
- Pages Pane
- 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.

25.7.2 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.
25.8  Tools

The **Tools** menu contains the following commands:

- **Global Resources**
- **Active Configuration**
- **User-Defined Tools**
- **Customize**
- **Restore Toolbars and Windows**
- **Options**

25.8.1  Global Resources

**Icon**

Displays the Global Resources dialog (*screenshot below*), in which you can:

- Specify the Global Resources XML File to use for global resources (*Definitions file*).
- Add file, folder, and database global resources (or aliases).
- Specify various configurations for each global resource (alias). Each configuration maps to a specific resource. (Edit a global resource to do this.)
- **Deploy to Server** enables you to deploy a global resource to a MobileTogether Server.
How to define global resources is described in detail in the section, Defining Global Resources.

25.8.2 Active Configuration

Placing the mouse over the command rolls out a submenu containing all the configurations defined in the currently active Global Resources XML File.

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.
25.8.3 User-Defined Tools

Placing the cursor over the User-defined Tools command rolls out a sub-menu containing custom-made commands that use external applications. In the screenshot below, two custom commands: (i) to open the parent folder in Windows Explorer, and (ii) to open the active file in Altova XMLSpy. You can create these commands in the Tools tab of the Customize dialog. 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.

25.8.4 Customize

The customize command lets you customize MobileTogether Designer to suit your personal needs.

25.8.4.1 Commands

The Commands tab enables you to add application commands to menus and toolbars according to your preference. Note that you cannot create new application commands or menus yourself.
To add a command to a toolbar or menu, do this:

1. Select the **All Commands** category in the **Categories** list box. The available commands appear in the **Commands** list box.
2. Click on a command in the **Commands** list box and drag it to an existing menu or toolbar. An I-beam appears when you place the cursor over a valid position to drop the command.
3. Release the mouse button at the position you want to insert the command.

Note the following:

- When you drag a command, a small button appears at the tip of mouse pointer: This indicates that the command is currently being dragged.
- An "x" below the pointer indicates that the command cannot be dropped at the current cursor position.
- If the cursor is moved to a position at which the command can be dropped (a toolbar or menu), the "x" disappears and an I-beam indicates the valid position.
- Commands can be placed in menus or toolbars. If you have created you own toolbar, 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.

25.8.4.2 Toolbars

Application toolbars contain icons for the most frequently used menu commands. Information about each icon is displayed in a tooltip and in the Status Bar when the cursor is placed over the icon. You can drag a toolbar to any location on the screen, where it will appear as a floating window.

The Toolbars tab enables you to do the following:

- Activate or deactivate specific toolbars (that is, to decide which ones to display in the interface)
- Set what icons are displayed in each toolbar
- Create your own specialized toolbars
The following functionality is available:

- **To activate or deactivate a toolbar**: Click its check box in the Toolbars list box.
- **To add a new toolbar**: Click the **New** button and give the toolbar a name in the Toolbar Name dialog that pops up. From the **Commands** 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).

### 25.8.4.3 Tools

The **Tools** tab allows you to set up commands to use external applications from within MobileTogether Designer. These commands will be added to the **Tools | User-defined Tools** 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.
25.8.4.4 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.

25.8.4.5 Menu
The Menu tab allows you to customize the two main menu bars (default and application menu bars) as well as context menus.

![Customize dialog](image)

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.
25.8.4.6 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.

25.8.5 Restore Toolbars and Windows

Shuts down MobileTogether Designer and restarts it with all toolbars and windows reset to the original state in which they were at installation.

25.8.6 Options

The **Options** command displays the Options dialog (*screenshot below*). The settings available in the various tabs are described in the sub-sections of this section.
The settings are organized into the following tabs:

- **General**
- **File**
- **Server Settings**
- **Trial Run on Client**
- **Simulation 1**
- **Simulation 2**
- **Email**
- **Network Proxy**
- **Java**
25.8.6.1 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**: In situations where designer input is required, you are prompted about whether to go ahead with the action or not. For example, when a shared page resource is modified, you are asked whether the modifications should be available on all pages that share the resource, or whether the modifications should apply to the current page only. You can also switch on/off notifications related to [placeholder controls](https://altova.com) and the [debugger](https://altova.com).
- **Videos**: An option to show a message about MobileTogether Designer demo videos when MobileTogether Designer is started with no design open. (To start MobileTogether Designer with no design open, close all designs and then close MobileTogether Designer.) The message contains a link to the demo video page on the Altova website. The videos on this page provide a quick introduction to the features of MobileTogether Designer.
- **Design View: Minimal column width**: Sets the minimal width of table columns in Design View. The slider provides options on a scale from 0 to 7. You can drag the slider, or use the **Move Left** and **Move Right** keys. The selected value does not affect the width of table columns on client devices.
25.8.6.2 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. To reload all page sources immediately, use the menu command **Project | Reload Page Source Structures**.
- **Cache validation**: Specifies when the cache is validated. Select the options you want.
25.8.6.3 Server Settings

In the Server Settings tab (screenshot below) you can define the connection and authentication settings of the MobileTogether Server to which you wish to connect MobileTogether Designer. These settings will be used when solutions are deployed to the server 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.

25.8.6.4 Trial Run on Client

In the Trial Run on Client tab (screenshot below) you can define the local port via which MobileTogether Designer connects with the client. For the Trial Run on Client feature, MobileTogether Designer itself acts as the MobileTogether Server and serves the design and related data files directly to the client.
SSL: Whether SSL is used.

Private key, certificate: Browse for the SSL private key and certificate (if SSL is used).

25.8.6.5 Simulation 1

In the Simulation 1 tab (screenshot below), you can specify default aspects of simulations: simulation checks and messages, the simulation language, whether comment and processing instructions should be shown in the simulator's XML data tree, etc. Additional simulation options are available in the next tab, Simulation 2.

Note: You can change simulation options before running a simulation by clicking the Simulator Options button in the Simulator's toolbar, and then changing settings as required. If you change a setting in the Simulator, that change is transmitted back to this Options pane and becomes the new default setting.
• **Simulation checks and messages**: Specifies whether warnings should be issued about (i) recursive sub pages, (ii) the keyboard being hidden on the client, (iii) when the client starts or stops receiving GPS information, (iv) PN management, (v) whether a dialog to set service triggers should be displayed.

• **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 language can be selected for local or server simulation by selecting the 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.
• **Options:** The settings you make here determine the default settings of the simulator. For example, you can set whether the simulator has access to the server or not. You can also specify whether a WiFi, LAN, or mobile network connection is simulated when the simulator starts. By default, the Simulate WiFi option is on. (In the Simulator, you can change the options at any time.) The Set Defaults button resets the options to their original default settings. You can also simulate that certain client-side apps (like a camera or calendar) can be accessed. After the simulator is started, you can change these settings in the simulator's toolbar or its Simulation menu. To set default data sources for some of these simulated client-side apps, go to the Simulation 2 tab in this dialog.

### 25.8.6.6 Simulation 2

In the Simulation 2 tab (screenshot below), you can specify simulation settings additional to those available in the Simulation 1 tab (see previous point). These settings include the definition of data sources for the simulation of client-side apps.
Menu Commands

Options

General | File | Server Settings | Trial Run on Client | Simulation 1 | Simulation 2 | Email | Network Proxy | Java

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\ ...

Geolocation Simulation
Geolocation Settings...

Contacts Simulation
- Use your Outlook Contacts
- Use a file that contains contacts information that can be used for simulation.

%PROGRAMFILES%\Altova\MobileTogetherDesigner5\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.xml ...

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

Read DB-Structure Simulation
A file that contains Read DB-Structure information that can be used for simulation.

C:\ Users\[user]\Documents\Altova\MobileTogetherDesigner5\DBReadStructure\Sample DB Read ...

Settings...

Server Variables Simulation
A file that contains Server Variable information that can be used for simulation.
First the Designer attempts to load the file mobiletogetherserver.cfg in the directory of the simulated solution. If that file does not exist the following file will be used:

%PROGRAMFILES%\Altova\MobileTogetherDesigner7\ServerVariables\Server Variables.cfg ...

In-App Purchase Simulation
A file that contains In-App purchase information that can be used for simulation.

%PROGRAMFILES%\Altova\MobileTogetherDesigner7\InAppPurchase\InAppPurchase Sample ...
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.

Geolocation Simulation
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 an XML 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 function.

Read DB-Structure Simulation
Specifies an XML file that contains a connection to a DB that is to be read for simulations of the DB Read Structure action. A dummy XML file, named Sample DB Read Structure.xml, is installed with MobileTogether Designer (version 5.0 and later) and is located in your Windows "My Documents" folder (click link below to see folder location). Note that this XML file contains no DB connection data when it is installed. But you can quickly generate connection information for multiple databases and save this information in the XML file as described below. The DB connection that is used during a simulation will be the one that has the same name as that of the connection in the DB Read Structure action. You can save the XML file to another location, or use some other XML file as the DB connection data file.

To generate one or more connections and save these to the currently selected XML file, do the following:

1. Click Settings.
2. In the dialog that appears (see screenshot below), click the Add DB button in the toolbar at top left.
3. In the **DB Connection Wizard** that now appears, add a new DB connection by following the wizard's steps.

4. After the DB connection has been created, it will appear in the list of connections (see screenshot above).

5. Click **OK** to save all the connections to the XML file. If you wish to remove a connection before saving all connections, select that connection in the dialog and click the **Delete** button in the toolbar at top left (see screenshot above).

**Server Variables Simulation**

You can specify a server configuration file (.cfg file) that contains the server variables you want to simulate. In a server configuration file, the server variables are stored in the **[Server Variables]** section, as shown in the snippet below:

```plaintext
[Server Variables]
Environment=Admin
Manual=AdminDocs
StartPage=Admin
```

The **mobiletogetherserver.cfg** file is located by default in the application data folder, the location of which is given below for each operating system. You can edit the .cfg configuration file in a text editor.

- **Linux**  
  `/var/opt/Altova/ExeNameUpperLower`

- **Mac**  
  `/var/Altova/ExeNameUpperLower`

- **Windows**  
  `C:\ProgramData\Altova\ExeNameUpperLower`

**In-App Purchase Simulation**

Provides data to simulate the product and purchase information that is available at app stores. See the topic about **simulating in-app purchases in MobileTogether Designer**.

* **Microsoft Outlook** is part of the Microsoft Office suite. Among other functionality, it provides a contacts manager and a calendar.*
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.
  
<table>
<thead>
<tr>
<th>Operating System</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7/8/10</td>
<td>C:\Users&lt;username&gt;\Documents</td>
</tr>
</tbody>
</table>

- **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.

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 7/8/10</td>
<td>C:\Program Files\Altova\</td>
</tr>
<tr>
<td>32-bit version on 64-bit OS</td>
<td>C:\Program Files (x86)\Altova\</td>
</tr>
</tbody>
</table>

### 25.8.6.7 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.

### 25.8.6.8 Network Proxy

The Network Proxy settings (*screenshot below*) enable you to configure custom proxy settings for MobileTogether Designer. The default is to use the system's proxy settings, so the settings will work without user intervention. If you wish to specify an alternative network proxy, use one of the options to define the setting you want.

**Use system proxy settings**

Uses the Internet Explorer (IE) settings configurable via the system proxy settings. It also queries the settings configured with `netsh.exe winhttp`.

Use the Internet Explorer (IE) settings configurable via the system proxy settings. It also queries the settings configured with `netsh.exe winhttp`. 
**Automatic proxy configuration**

The following options are provided:

- **Auto-detect settings**: Looks up a WPAD script (http://wpad.LOCALDOMAIN/wpad.dat) via DHCP or DNS, and uses this script for proxy setup.
- **Script URL**: Specify an HTTP URL to a proxy-auto-configuration (.pac) script that is to be used for proxy setup.
- **Reload**: Resets and reloads the current auto-proxy-configuration. This action requires Windows 8 or newer, and may need up to 30s to take effect.

**Manual proxy configuration**

Manually specify the fully qualified host name and port for the proxies of the respective protocols. A supported scheme may be included in the host name (for example: http://hostname). It is not required that the scheme is the same as the respective protocol if the proxy supports the scheme.

The following options are provided:

- **Use this proxy for all protocols**: Uses the host name and port of the HTTP Proxy for all protocols.
- **No Proxy for**: A semi-colon (;) separated list of fully qualified host names, domain names, or IP addresses for hosts that should be used without a proxy. IP addresses may not be truncated and IPv6 addresses have to be enclosed by square brackets (for example: [2606:2800:220:1:248:1893:25c8:1946]). Domain names must start with a leading dot (for example: .example.com).
- **Do not use the proxy server for local addresses**: If checked, adds <local> to the No Proxy for list. If this option is selected, then the following will not use the proxy: (i) 127.0.0.1, (ii) [:1], (iii) all host names not containing a dot character (\).

**Current proxy settings**

Provides a verbose log of the proxy detection. It can be refreshed with the Refresh button to the right of the Test URL field (for example, when changing the test URL, or when the proxy settings have been changed).

- **Test URL**: A test URL can be used to see which proxy is used for that specific URL. No I/O is done with this URL. This field must not be empty if proxy-auto-configuration is used (either through Use system proxy settings or Authomatic proxy configuration).

**25.8.6.9 Java**

On the Java tab, you can optionally enter the path to a Java VM (Virtual Machine) on your file system. Note that adding a custom Java VM path is not always necessary. By default, MobileTogether Designer attempts to detect the Java VM path automatically by reading (in this order) the Windows registry and the JAVA_HOME environment variable. The custom path added on this dialog box will take priority over any other Java VM path detected automatically.

You may need to add a custom Java VM path, for example, if you are using a Java virtual machine which does not have an installer and does not create registry entries (for example, Oracle's OpenJDK). You might also want to set this path if you need to override, for whatever reason, any Java VM path detected automatically by MobileTogether Designer.
Note the following:

- The Java VM path is shared between Altova desktop (not server) applications. Consequently, if you change it in one application, it will automatically apply to all other Altova applications.
- The path must point to the `jvm.dll` file from the `bin\server` or `bin\client` directory, relative to the directory where the JDK was installed.
- The MobileTogether Designer platform (32-bit, 64-bit) must be the same as that of the JDK.
- After changing the Java VM path, you may need to restart MobileTogether Designer for the new settings to take effect.

Changing the Java VM path affects database connectivity via JDBC.
25.9 Window

The Window menu contains the following commands:

- Cascade
- Tile Horizontally
- Tile Vertically
- Close
- Close All
- Close All But Active
- Currently Open Windows List

25.9.1 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.

25.9.2 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.
25.9.3 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.
25.10 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

25.10.1 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 (i) press Enter or (ii) click List Topics. 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.
25.10.2 Activation, Order Form, Registration, Updates

Software Activation

- Description

After you download your Altova product software, you can license—or activate—it using either a free evaluation key or a purchased permanent license key.

- **Free permanent license.** When you first start the software after downloading and installing it, the Software Activation dialog will pop up. In it is a button to request a free permanent license. Enter your name, company, and e-mail address in the dialog that appears, and click Request. A license file is sent to the e-mail address you entered and should reach you in a few minutes. Save the license file to a suitable location. When you clicked Request, an entry field appeared at the bottom of the Request dialog. This field takes the path to the license file. Browse for or enter the path to the license file, and click OK. (In the Software Activation dialog, you can also click Upload a New License to access a dialog in which the path to the license file is entered.)

**Note:** For multi-user licenses, each user will be prompted to enter his or her own name.

```plaintext
Your license email and the different ways to license (activate) your Altova product

The license email that you receive from Altova will contain your license file as an attachment. The license file has a .altova_licenses file extension.

To activate your Altova product, you can do one of the following:

- Save the license file (.altova_licenses) to a suitable location, double-click the license file, enter any requested details in the dialog that appears, and finish by clicking Apply Keys.
- Save the license file (.altova_licenses) to a suitable location. In your Altova product, select the menu command Help | Software Activation, and then Upload a New License. Browse for or enter the path to the license file, and click OK.
```

The Software Activation dialog can be accessed at any time by clicking the Help | Software Activation command.

Order Form

- Description

The Order Form command takes you to the secure Altova Online Shop, where you can purchase license keys for Altova products. Altova MobileTogether Designer, however, is a free product and does not need to be ordered. You can request a free permanent license directly from within the program, via the Software Activation dialog (see above).

Registration
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.

25.10.3 Other Commands

▼ Support Center

Description

A link to the MobileTogether Support Forum on the Internet. The Support Forums provides discussion forums, where problems are discussed, and announcements related to MobileTogether.

▼ Show Video Demos

Description

A link to the MobileTogether Designer video demo page on the Altova website. The videos on this page show you how to get started with using MobileTogether Designer.

▼ MobileTogether Designer on the Internet

Description

A link to the Altova website on the Internet. You can learn more about MobileTogether Designer and related technologies and products at the Altova website.

▼ About MobileTogether Designer

Description

Displays the splash window and version number of your product.
26 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:

<table>
<thead>
<tr>
<th>Browser</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE 8.0+</td>
<td>Firefox 3.5+</td>
</tr>
</tbody>
</table>
Appendices

Information included in this section:

- [XSLT and XPath/XQuery Functions](#)
- [License Information](#)
27.1  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 timezones of the values being compared need to be known. When the timezone is not explicitly given in such a value, the implicit timezone is used. The implicit timezone is taken from the system clock, and its value can be checked with the implicit-timezone() function.

Collations
The default collation is the Unicode codepoint collation, which compares strings on the basis of their Unicode codepoint. The engine uses the Unicode Collation Algorithm. Other supported collations are the ICU collations listed below; to use one of these, supply its URI as given in the table below. Any string comparisons, including for the max and min functions, will be made according to the specified collation. If the collation option is not specified, the default Unicode-codepoint collation is used.

<table>
<thead>
<tr>
<th>Language</th>
<th>URIs</th>
</tr>
</thead>
<tbody>
<tr>
<td>da: Danish</td>
<td>da_DK</td>
</tr>
<tr>
<td>de: German</td>
<td>de_AT, de_BE, de_CH, de_DE, de_LI, de_LU</td>
</tr>
</tbody>
</table>
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.

### 27.1.1 Altova Extension Functions

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of `[altova:]`. For example: `add-years-to-date [altova:]`.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

<table>
<thead>
<tr>
<th>Language</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>es: Spanish</td>
<td>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</td>
</tr>
<tr>
<td>it: Italian</td>
<td>it_CH, it_IT</td>
</tr>
<tr>
<td>ja: Japanese</td>
<td>ja_JP</td>
</tr>
<tr>
<td>nb: Norwegian Bokmal</td>
<td>nb_NO</td>
</tr>
<tr>
<td>nl: Dutch</td>
<td>nl_AW, nl_BE, nl_NL</td>
</tr>
<tr>
<td>nn: Nynorsk</td>
<td>nn_NO</td>
</tr>
<tr>
<td>ru: Russian</td>
<td>ru_MD, ru_RU, ru_UA</td>
</tr>
<tr>
<td>sv: Swedish</td>
<td>sv_FI, sv_SE</td>
</tr>
</tbody>
</table>

**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**
27.1.1.1 XPath/XQuery Functions: Date and Time

Altova’s date/time extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data held as XML Schema’s various date and time datatypes.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of [altova:]. For example: add-years-to-date [altova:].
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function without any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: add-years-to-date(xs:date("2014-01-15"), 10).

<table>
<thead>
<tr>
<th>XPath functions (used in XPath expressions in XSLT):</th>
<th>XP1</th>
<th>XP2</th>
<th>XP3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSLT functions (used in XPath expressions in XSLT):</td>
<td>XSLT1</td>
<td>XSLT2</td>
<td>XSLT3</td>
</tr>
<tr>
<td>XQuery functions (used in XQuery expressions in XQuery):</td>
<td>XQ1</td>
<td>XQ3.1</td>
<td></td>
</tr>
</tbody>
</table>

Grouped by functionality

- Add a duration to xs:dateTime and return xs:dateTime
- Add a duration to xs:dateTime and return xs:dateTime
- Add a duration to xs:dateTime and return xs:dateTime
- Add a duration to xs:dateTime and return xs:dateTime
- Format and retrieve durations
- Remove timezone from functions that generate current dateTime
- Return days, hours, minutes, and seconds from durations
- 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
Add a duration to xs:dateTime

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:]

```xml
altova:add-years-to-dateTime(DateTime as xs:dateTime, Years as xs:integer) as xs:dateTime
```

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:]

altova:add-months-to-dateTime(DateTime as xs:dateTime, Months as xs:integer) as xs:dateTime

This function adds a duration in months to a date. 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.
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 2014-01-05T14:00:00
- 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("2014-01-15T14:00:00"), 45) returns 2014-01-15T14:55:00
- add-minutes-to-dateTime(xs:dateTime("2014-01-15T14:00:00"), -5) returns 2014-01-
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`

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"), -2)` returns `2014-01-13`
• 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

Format and retrieve durations  XP3.1  XQ3.1
These functions parse an input xs:duration or xs:string and return, respectively, an xs:string or xs:duration.

▼ format-duration [altova:]
format-duration(Duration as xs:duration, Picture as xs:string) as xs:string  XP3.1  XQ3.1
Formats a duration, which is submitted as the first argument, according to a picture string submitted as the second argument. The output is a text string formatted according to the picture string.

Examples
• format-duration(xs:duration("P2DT2H53M11.7S"), "Days:[D01] Hours:[H01] Minutes: [m01] Seconds:[s01] Fractions:[f0]") returns "Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"
• format-duration(xs:duration("P3M2DT2H53M11.7S"), "Months:[M01] Days:[D01] Hours: [H01] Minutes:[m01]") returns "Months:03 Days:02 Hours:02 Minutes:53"

▼ parse-duration [altova:]
parse-duration(InputString as xs:string, Picture as xs:string) as xs:duration  XP3.1  XQ3.1
Takes a patterned string as the first argument, and a picture string as the second argument. The input string is parsed on the basis of the picture string, and an xs:duration is returned.

Examples
• parse-duration("Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7"), "Days:[D01] Hours:[H01] Minutes:[m01] Seconds:[s01] Fractions:[f0]") returns "P2DT2H53M11.7S"
• parse-duration("Months:03 Days:02 Hours:02 Minutes:53 Seconds:11 Fractions:7", "Months:[M01] Days:[D01] Hours:[H01] Minutes:[m01]") returns "P3M2DT2H53M"

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`

---

**Remove the timezone part from date/time datatypes**  **XP3.1** **XQ3.1**

These functions remove the timezone from the current **xs:dateTime**, **xs:date**, or **xs:time** values, respectively. Note that the difference between **xs:dateTime** and **xs:dateTimeStamp** is that in the case of the latter the timezone part is required (while it is optional in the case of the former). So the format of an **xs:dateTimeStamp** value is: CCYY-MM-DDThh:mm:ss.sss±hh:mm or CCYY-MM-DDThh:mm:ss.sssZ. If the date and time is read from the system clock as **xs:dateTimeStamp**, the **current-dateTime-no-TZ()** function can be used to remove the timezone if so required.

**current-date-no-TZ [altova:]**

`current-date-no-TZ() as xs:date` **XP3.1** **XQ3.1**

This function takes no argument. It removes the timezone part of **current-date()** (which is the current date according to the system clock) and returns an **xs:date** value.

- **Examples**
  - If the current date is **2014-01-15+01:00**:
    - `current-date-no-TZ()` returns **2014-01-15**
current-dateTime-no-TZ [altova:]  

```
current-dateTime-no-TZ() as xs:dateTime XP3.1 XQ3.1
```

This function takes no argument. It removes the timezone part of `current-dateTime()` (which is the current date-and-time according to the system clock) and returns an `xs:dateTime` value.

**Examples**

If the current dateTime is `2014-01-15T14:00:00+01:00`:

- `current-dateTime-no-TZ()` returns `2014-01-15T14:00:00`

current-time-no-TZ [altova:]  

```
current-time-no-TZ() as xs:time XP3.1 XQ3.1
```

This function takes no argument. It removes the timezone part of `current-time()` (which is the current time according to the system clock) and returns an `xs:time` value.

**Examples**

If the current time is `14:00:00+01:00`:

- `current-time-no-TZ()` returns `14:00:00`

date-no-TZ [altova:]  

```
date-no-TZ(InputDate as xs:date) as xs:date XP3.1 XQ3.1
```

This function takes an `xs:date` argument, removes the timezone part from it, and returns an `xs:date` value. Note that the date is not modified.

**Examples**

- `date-no-TZ(xs:date("2014-01-15+01:00"))` returns `2014-01-15`

dateTime-no-TZ [altova:]  

```
dateTime-no-TZ(InputDateTime as xs:dateTime) as xs:dateTime XP3.1 XQ3.1
```

This function takes an `xs:dateTime` argument, removes the timezone part from it, and returns an `xs:dateTime` value. Note that neither the date nor the time is modified.

**Examples**

- `dateTime-no-TZ(xs:dateTime("2014-01-15T14:00:00+01:00"))` returns `2014-01-15T14:00:00`

time-no-TZ [altova:]  

```
time-no-TZ(InputTime as xs:time) as xs:time XP3.1 XQ3.1
```

This function takes an `xs:time` argument, removes the timezone part from it, and returns an `xs:time` value. Note that the time is not modified.

**Examples**

- `time-no-TZ(xs:time("14:00:00+01:00"))` returns `14:00:00`
Return the number of days, hours, minutes, seconds from durations XP3.1 XQ3.1

These functions return the number of days in a month, and the number of hours, minutes, and seconds, respectively, from durations.

▼ days-in-month [altova:]

```
days-in-month(Year as xs:integer, Month as xs:integer) as xs:integer XP3.1 XQ3.1
```

Returns the number of days in the specified month. The month is specified by means of the Year and Month arguments.

**Examples**
- `days-in-month(2018, 10)` returns 31
- `days-in-month(2018, 2)` returns 28
- `days-in-month(2020, 2)` returns 29

▼ hours-from-dayTimeDuration-accumulated

```
hours-from-dayTimeDuration-accumulated(DayAndTime as xs:duration) as xs:integer XP3.1 XQ3.1
```

Returns the total number of hours in the duration submitted by the DayAndTime argument (which is of type `xs:duration`). The hours in the Day and Time components are added together to give a result that is an integer. A new hour is counted only for a full 60 minutes. Negative durations result in a negative hour value.

**Examples**
- `hours-from-dayTimeDuration-accumulated(xs:duration("PT60M"))` returns 60, which is the total number of hours in 1 hour.
- `hours-from-dayTimeDuration-accumulated(xs:duration("PT1H"))` returns 60, which is the total number of hours in 1 hour.
- `hours-from-dayTimeDuration-accumulated(xs:duration("PT1H60M"))` returns 60, which is the total number of hours in 1 hour and 1 minute.
- `hours-from-dayTimeDuration-accumulated(xs:duration("PT1H119M"))` returns 60, which is the total number of hours in 1 hour and 1 minute and 19 seconds.
- `hours-from-dayTimeDuration-accumulated(xs:duration("PT1H120M"))` returns 60, which is the total number of hours in 1 hour and 1 minute and 20 seconds.
- `hours-from-dayTimeDuration-accumulated(xs:duration("-PT1H"))` returns -60, which is the total number of hours in negative 1 hour.

▼ minutes-from-dayTimeDuration-accumulated

```
minutes-from-dayTimeDuration-accumulated(DayAndTime as xs:duration) as xs:integer XP3.1 XQ3.1
```

Returns the total number of minutes in the duration submitted by the DayAndTime argument (which is of type `xs:duration`). The minutes in the Day and Time components are added together to give a result that is an integer. Negative durations result in a negative minute value.

**Examples**
- `minutes-from-dayTimeDuration-accumulated(xs:duration("PT60M"))` returns 60, which is the total number of minutes in 1 hour.
- `minutes-from-dayTimeDuration-accumulated(xs:duration("PT1H"))` returns 60, which is the total number of minutes in 1 hour.
total number of minutes in 1 hour.

- \texttt{minutes-from-dayTimeDuration-accumulated(xs:duration("PT1H40M"))} returns 100
- \texttt{minutes-from-dayTimeDuration-accumulated(xs:duration("P1D"))} returns 1440, which is the total number of minutes in 1 day.
- \texttt{minutes-from-dayTimeDuration-accumulated(xs:duration("-P1DT60M"))} returns -1500

\begin{itemize}
\item \texttt{seconds-from-dayTimeDuration-accumulated(DayAndTime as xs:duration)} as \texttt{xs:integer}
\end{itemize}

\noindent Returns the total number of seconds in the duration submitted by the \texttt{DayAndTime} argument (which is of type \texttt{xs:duration}). The seconds in the \texttt{Day} and \texttt{Time} components are added together to give a result that is an integer. Negative durations result in a negative seconds value.

\begin{itemize}
\item \texttt{seconds-from-dayTimeDuration-accumulated(xs:duration("PT1M"))} returns 60, which is the total number of seconds in 1 minute.
\item \texttt{seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H"))} returns 3600, which is the total number of seconds in 1 hour.
\item \texttt{seconds-from-dayTimeDuration-accumulated(xs:duration("PT1H2M"))} returns 3720
\item \texttt{seconds-from-dayTimeDuration-accumulated(xs:duration("P1D"))} returns 86400, which is the total number of seconds in 1 day.
\item \texttt{seconds-from-dayTimeDuration-accumulated(xs:duration("-P1DT1M"))} returns -86460
\end{itemize}

\underline{Return the weekday from \texttt{xs:dateTime} or \texttt{xs:date}}

\begin{itemize}
\item \texttt{weekday-from-dateTime(DateTime as xs:dateTime, Format as xs:integer)} as \texttt{xs:integer}
\end{itemize}

\noindent These functions return the weekday (as an integer) from \texttt{xs:dateTime} or \texttt{xs:date}. The days of the week are numbered (using the American format) from 1 to 7, with \texttt{Sunday}=1. In the European format, the week starts with \texttt{Monday} (=1). The American format, where \texttt{Sunday}=1, can be set by using the integer 0 where an integer is accepted to indicate the format.

\begin{itemize}
\item \texttt{weekday-from-dateTime(DateTime as xs:dateTime("2014-02-03T09:00:00"))} returns 2, which would indicate a Monday.
\end{itemize}
Examples

- \texttt{weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 1)} returns 1, which would indicate a Monday
- \texttt{weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 4)} returns 1, which would indicate a Monday
- \texttt{weekday-from-dateTime(xs:dateTime("2014-02-03T09:00:00"), 0)} returns 2, which would indicate a Monday.

\textbf{weekday-from-date} [altova:]

\texttt{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 \textit{(see next signature below)}.

Examples

- \texttt{weekday-from-date(xs:date("2014-02-03+01:00"))} returns 2, which would indicate a Monday.

\texttt{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 \texttt{(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 \textit{(see previous signature)}.

Examples

- \texttt{weekday-from-date(xs:date("2014-02-03"), 1)} returns 1, which would indicate a Monday
- \texttt{weekday-from-date(xs:date("2014-02-03"), 4)} returns 1, which would indicate a Monday
- \texttt{weekday-from-date(xs:date("2014-02-03"), 0)} returns 2, which would indicate a Monday.

Return the week number from \texttt{xs:dateTime} or \texttt{xs:date} \texttt{XP2 XQ1 XP3.1 XQ3.1}

These functions return the week number (as an integer) from \texttt{xs:dateTime} or \texttt{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).

\textbf{weeknumber-from-date} [altova:]

\texttt{weeknumber-from-date(Date as xs:date, Calendar as xs:integer) as xs:integer XP3.1 XQ3.1}

Returns the week number of the submitted \texttt{Date} argument as an integer. The second argument \texttt{(Calendar)} specifies the calendar system to follow. Supported \texttt{Calendar} values are:

- 0 = US calendar \textit{(week starts Sunday)}
- 1 = ISO standard, European calendar \textit{(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
```

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.

---

**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
```

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:]

```xquery
build-time(Hours as xs:integer, Minutes as xs:integer, Seconds as xs:integer) as xs:time
```

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

```xquery
build-time(Hours as xs:integer, Minutes as xs:integer, Seconds as xs:integer, TimeZone as xs:string) as xs:time
```

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:]

```xquery
build-duration(Years as xs:integer, Months as xs:integer) as xs:yearMonthDuration
```

Takes two arguments to build a value of type `xs:yearMonthDuration`. The first argument provides the Years part of the duration value, while the second argument provides the Months part. If the second (Months) argument is greater than or equal to 12, then the integer is divided by 12; the quotient is added to the first argument to provide the Years part of the duration value while the remainder (of the division) provides the Months part. To build a duration of type `xs:dayTimeDuration`, see the next signature.

**Examples**
- `build-duration(2, 10)` returns P2Y10M
- `build-duration(14, 27)` returns P16Y3M
- `build-duration(2, 24)` returns P4Y

```xquery
build-duration(Days as xs:integer, Hours as xs:integer, Minutes as xs:integer, Seconds as xs:integer) as xs:dayTimeDuration
```

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`

Construct date, dateTime, and time datatypes from string input

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:]**

```xml
parse-date(Date as xs:string, DatePattern as xs:string) as xs:date
```

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.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Date</td>
</tr>
<tr>
<td>M</td>
<td>Month</td>
</tr>
<tr>
<td>Y</td>
<td>Year</td>
</tr>
</tbody>
</table>

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"), "]D]-[M]-[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:]**

```xml
parse-dateTime(DateTime as xs:string, DateTimePattern as xs:string) as xs:dateTime
```

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.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Date</td>
</tr>
<tr>
<td>M</td>
<td>Month</td>
</tr>
<tr>
<td>Y</td>
<td>Year</td>
</tr>
</tbody>
</table>
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:]`

```xml
parse-time(Time as xs:string, TimePattern as xs:string) as xs:time
```

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`

### Age-related functions

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:]`

```xml
age(StartDate as xs:date) as xs:integer
```

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:

- \( \text{age}(\text{xs:date("2013-01-15"))} \) returns 1
- \( \text{age}(\text{xs:date("2013-01-16"))} \) returns 0
- \( \text{age}(\text{xs:date("2015-01-15"))} \) returns -1
- \( \text{age}(\text{xs:date("2015-01-14"))} \) returns 0

\[
\text{age}(\text{Start Date} \text{ as xs:date, End Date as xs:date) as xs:integer} \]

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:

- \( \text{age}(\text{xs:date("2000-01-15"), xs:date("2010-01-15")}) \) returns 10
- \( \text{age}(\text{xs:date("2000-01-15"), current-date()} \) returns 14 if the current date is 2014-01-15
- \( \text{age}(\text{xs:date("2014-01-15"), xs:date("2010-01-15")}) \) returns -4

\[
\text{age-details [altova:]}
\]

\[
\text{age-details(Input Date as xs:date) as (xs:integer)*} \]

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:

- \( \text{age-details}(\text{xs:date("2014-01-16")}) \) returns (0 0 1)
- \( \text{age-details}(\text{xs:date("2014-01-14")}) \) returns (0 0 1)
- \( \text{age-details}(\text{xs:date("2013-01-16")}) \) returns (1 0 1)
- \( \text{age-details}(\text{current-date()}) \) returns (0 0 0)

\[
\text{age-details(Date-1 as xs:date, Date-2 as xs:date) as (xs:integer)*} \]

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**

- \( \text{age-details}(\text{xs:date("2014-01-16")}, \text{xs:date("2014-01-15")}) \) returns (0 0 1)
- \( \text{age-details}(\text{xs:date("2014-01-15")}, \text{xs:date("2014-01-16")}) \) returns (0 0 1)
27.1.1.2 XPath/XQuery Functions: Geolocation

The following geolocation XPath/XQuery extension functions are supported in the current version of MobileTogether Designer.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of [altova:]. For example: `add-years-to-date [altova:]`.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function without any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

<table>
<thead>
<tr>
<th>XPath functions (used in XPath expressions in XSLT):</th>
<th>XP1</th>
<th>XP2</th>
<th>XP3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSLT functions (used in XPath expressions in XSLT):</td>
<td>XSLT1</td>
<td>XSLT2</td>
<td>XSLT3</td>
</tr>
<tr>
<td>XQuery functions (used in XQuery expressions in XQuery):</td>
<td>XQ1</td>
<td>XQ3.1</td>
<td></td>
</tr>
</tbody>
</table>

format-geolocation [altova:]

`format-geolocation(Latitude as xs:decimal, Longitude as xs:decimal, GeolocationOutputStringFormat as xs:integer) as xs:string XP3.1 XQ3.1`

Takes the latitude and longitude as the first two arguments, and outputs the geolocation as a string. The third argument, `GeolocationOutputStringFormat`, is the format of the geolocation output string; it uses integer values from 1 to 4 to identify the output string format (see ‘Geolocation output string formats’ below). Latitude values range from +90 to -90 (N to S). Longitude values range from +180 to -180 (E to W).

Note: The `image-exif-data` function and the Exif metadata’s attributes can be used to supply the input strings.

Examples

- `format-geolocation(33.33, -22.22, 4)` returns the `xs:string "33.33 -22.22"`
- `format-geolocation(33.33, -22.22, 2)` returns the `xs:string "33.33N 22.22W"`
- `format-geolocation(-33.33, 22.22, 2)` returns the `xs:string "33.33S 22.22E"`
- `format-geolocation(33.33, -22.22, 1)` returns the `xs:string "33°19'48.00"S 22°13'12.00"E"`

Geolocation output string formats:

The supplied latitude and longitude is formatted in one of the output formats given below. The desired format is identified by its integer ID (1 to 4). Latitude values range from +90 to -90 (N to S). Longitude values range from +180 to -180 (E to W).
1

**Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)**

\[D°M'S.SS"N/S \quad D°M'S.SS"E/W\]

*Example:* \(33°55'11.11"N \quad 22°44'66.66"W\)

2

**Decimal degrees, with suffixed orientation (N/S, E/W)**

\[D.DDN/S \quad D.DDE/W\]

*Example:* \(33.33N \quad 22.22W\)

3

**Degrees, minutes, decimal seconds, with prefixed sign (+/−); plus sign for (N/E) is optional**

\[+/−D°M'S.SS" \quad +/−D°M'S.SS"\]

*Example:* \(33°55'11.11" \quad −22°44'66.66"\)

4

**Decimal degrees, with prefixed sign (+/−); plus sign for (N/E) is optional**

\[+/−D.DD \quad +/−D.DD\]

*Example:* \(33.33 \quad −22.22\)

**Altova Exif Attribute: Geolocation**

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, with units added (see table below).

<table>
<thead>
<tr>
<th>GPSLatitude</th>
<th>GPSLatitudeRef</th>
<th>GPSLongitude</th>
<th>GPSLongitudeRef</th>
<th>Geolocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>51</td>
<td>151</td>
<td>11.73</td>
<td>(33°51'21.91&quot;S) 151°13'11.73&quot;E</td>
</tr>
</tbody>
</table>

**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"W")` returns the sequence of two `xs:decimals` (48.858222222222, 24.2945)
- `parse-geolocation("48°51.6"N 24°17.4"W")` returns the sequence of two `xs:decimals` (48.858222222222, 24.2945)
- `parse-geolocation( image-exif-data(//MyImages/Image20141130.01)/@Geolocation )` returns a sequence of two `xs:decimals`

Geolocation input string formats:

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from +90 to −90 (N to S). Longitude values range from +180 to −180 (E to W).

Note: If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow (" ) while unit indicators that are escaped are highlighted in blue ("').

- Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)
  \[D°M'S.S'SS"N/S  D°M'S.S'SS"W/E\]
  Example: 33°55'11.11"N  22°44'55.25"W

- Degrees, minutes, decimal seconds, with prefixed sign (+/-); the plus sign for (N/E) is optional
  \[+/-D°M'S.S'SS"  +/-D°M'S.S'SS"\]
  Example: 33°55'11.11"  -22°44'55.25"

- Degrees, decimal minutes, with suffixed orientation (N/S, E/W)
  \[D°M.MM'N/S  D°M.MM'W/E\]
  Example: 33°55.55'N  22°44.44'W

- Degrees, decimal minutes, with prefixed sign (+/-); the plus sign for (N/E) is optional
  \[+/-D°M.MM'  +/-D°M.MM'\]
  Example: +33°55.55'  -22°44.44'

- Decimal degrees, with suffixed orientation (N/S, E/W)
  \[D.DDDN/S  D.DDWE\]
  Example: 33.33N  22.22W

- Decimal degrees, with prefixed sign (+/-); the plus sign for (N/S E/W) is optional
  \[+/-D.DD  +/-D.DD\]
  Example: 33.33  -22.22

Examples of format-combinations:

33.33N  -22°44'55.25"
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`, and `GPSLongitudeRef`, with units added (see table below).

<table>
<thead>
<tr>
<th>GPSLatitude</th>
<th>GPSLatitudeRef</th>
<th>GPSLongitude</th>
<th>GPSLongitudeRef</th>
<th>Geolocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>33°51'21.91&quot;</td>
<td>S</td>
<td>151°13'11.73&quot;</td>
<td>E</td>
<td>33°51'21.91&quot;S 151°13'11.73&quot;E</td>
</tr>
</tbody>
</table>

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 (”).

### Examples

- **geolocation-distance-km**

  `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.

- **Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)**

  D°M'S.SS"N/S  D°M'S.SS"W/E

  **Example:** 33°55'11.11"N  22°44'55.25"W

- **Degrees, minutes, decimal seconds, with prefixed sign (+/-); the plus sign for (N/E) is**
optional
+/-D°M.MM'SS'' +/–D°M.MM'SS''

*Example:* 33°55'11.11"  -22°44'55.25"

- Degrees, decimal minutes, with suffixed orientation (N/S, E/W)
  D°M.MM'N/S  D°M.MM'W/E
  *Example:* 33°55.55'N  22°44.44'W

- Degrees, decimal minutes, with prefixed sign (+/-); the plus sign for (N/E) is optional
  +/-D°M.MM'  +/-D°M.MM'
  *Example:* +33°55.55'  -22°44.44'

- Decimal degrees, with suffixed orientation (N/S, E/W)
  D.DD/N/S  D.DD/W/E
  *Example:* 33.33N  22.22W

- Decimal degrees, with prefixed sign (+/-); the plus sign for (N/S E/W) is optional
  +/-D.DD  +/-D.DD
  *Example:* 33.33  -22.22

*Examples of format-combinations:*
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45

**Altova Exif Attribute: Geolocation**

The Altova XPath/XQuery Engine generates the custom attribute `Geolocation` from standard Exif metadata tags. `Geolocation` is a concatenation of four Exif tags: `GPSLatitude`, `GPSLatitudeRef`, `GPSLongitude`, `GPSLongitudeRef`, with units added *see table below*.

<table>
<thead>
<tr>
<th>GPSLatitude</th>
<th>GPSLatitudeRef</th>
<th>GPSLongitude</th>
<th>GPSLongitudeRef</th>
<th>Geolocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>51</td>
<td>151</td>
<td>13</td>
<td>33°51'21.91&quot;S 151°13'11.73&quot;E</td>
</tr>
</tbody>
</table>

**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**
\textbf{geolocation-distance-mi}(33.33, -22.22, 48°51'29.6"N, 24°17'40.2"W) returns the \texttt{xs:decimal 2599.40652340653}.

\textit{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).

\textbf{Note}: If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow ("") while unit indicators that are escaped are highlighted in blue ("'").

- Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)
  \[D°M'S.SS"N/S D°M'S.SS"W/E\]
  \textit{Example}: 33°55'11.11"N 22°44'55.25"W

- Degrees, minutes, decimal seconds, with prefixed sign (+/-); the plus sign for (N/E) is optional
  \[+/-D°M'S.SS" +/D°M'S.SS"\]
  \textit{Example}: 33°55'11.11" -22°44'55.25"W

- Degrees, decimal minutes, with suffixed orientation (N/S, E/W)
  \[D°M.MM'N/S D°M.MM'W/E\]
  \textit{Example}: 33°55.55'N 22°44.44'W

- Degrees, decimal minutes, with prefixed sign (+/-); the plus sign for (N/E) is optional
  \[+/-D°M.MM' +/D°M.MM'\]
  \textit{Example}: +33°55.55' -22°44.44'

- Decimal degrees, with suffixed orientation (N/S, E/W)
  \[D.DDN/S D.DDW/E\]
  \textit{Example}: 33.33N 22.22W

- Decimal degrees, with prefixed sign (+/-); the plus sign for (N/S, E/W) is optional
  \[+/-D.DD +/D.DD\]
  \textit{Example}: 33.33 -22.22

\textit{Examples of format-combinations:}

- 33.33N -22°44'55.25"W
- 33.33 22°44'55.25"W
- 33.33 22.45

\textit{Altova Exif Attribute: Geolocation}

The Altova XPath/XQuery Engine generates the custom attribute \texttt{Geolocation} from standard Exif metadata tags. \texttt{Geolocation} is a concatenation of four Exif tags: \texttt{GPSLatitude}, \texttt{GPSLatitudeRef}, \texttt{GPSLongitude}, \texttt{GPSLongitudeRef}, with units added \textit{(see table below)}.
geolocations-bounding-rectangle [altova:]

geolocations-bounding-rectangle (Geolocations as xs:sequence,
GeolocationOutputStringFormat as xs:integer) as xs:string XP3.1 XQ3.1

Takes a sequence of strings as its first argument; each string in the sequence is a geolocation. The
function returns a sequence of two strings which are, respectively, the top-left and bottom-right
gеolocation coordinates of a bounding rectangle that is optimally sized to enclose all the
gеолocations submitted in the first argument. The formats in which a geolocation input string can be supplied are listed below (see
'Geolocation input string formats'). Latitude values range from +90 to −90 (N to S). Longitude values range from +180 to −180 (E to W).

The function’s second argument specifies the format of the two geolocation strings in the output sequence.
The argument takes an integer value from 1 to 4, where each value identifies a different geolocation string
format (see 'Geolocation output string formats' below).

Note: The image-exif-data function and the Exif metadata’s attributes can be used to supply the
input strings.

Examples

- geolocations-bounding-rectangle((Roles-geoloc in "48.2143531 16.3707266", "51.50939 -0.11832"),
1) returns the sequence ("51°30'33.804"N 0°7'5.952"W", "48°12'51.67116"N 16°
22'14.61576"E")
- geolocations-bounding-rectangle((Roles-geoloc in "48.2143531 16.3707266", "51.50939 -0.11832",
"42.5584577 -70.8893334"), 4) returns the sequence ("51.50939 -70.8893334",
"42.5584577 16.3707266")

Geolocation input string formats:
The geolocation input string must contain latitude and longitude (in that order) separated by
whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can
be in one format and longitude can be in another. Latitude values range from +90 to −90 (N to S). Longitude values range from +180 to −180 (E to W).

Note: If single quotes or double quotes are used to delimit the input string argument, this will create
a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-
values and second-values. In such cases, the quotes that are used for indicating minute-values and
second-values must be escaped by doubling them. In the examples in this section, quotes used to
delimit the input string are highlighted in yellow ("), while unit indicators that are escaped are
highlighted in blue ("').

- Degrees, minutes, decimal seconds, with suffixed orientation (N/S; E/W)
### Geolocation output string formats:

The supplied latitude and longitude is formatted in one of the output formats given below. The desired format is identified by its integer ID (1 to 4). Latitude values range from +90 to -90 (N to S). Longitude values range from +180 to -180 (E to W).

<table>
<thead>
<tr>
<th>Format ID</th>
<th>Format Description</th>
<th>Example Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)</td>
<td>33°55'11.11&quot;N  22°44'55.25&quot;W</td>
</tr>
<tr>
<td>2</td>
<td>Decimal degrees, with suffixed orientation (N/S, E/W)</td>
<td>33.33N  22.22W</td>
</tr>
<tr>
<td>3</td>
<td>Degrees, minutes, decimal seconds, with prefixed sign (+/-); plus sign for (N/E) is optional</td>
<td>33°55'11.11&quot;  -22°44'66.66&quot;</td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th>GPSLatitude</th>
<th>GPSLatitudeRef</th>
<th>GPSLongitude</th>
<th>GPSLongitudeRef</th>
<th>Geolocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 51 21.91</td>
<td>S</td>
<td>151 13 11.73</td>
<td>E</td>
<td>33°51'21.91&quot;S 151°13'11.73&quot;E</td>
</tr>
</tbody>
</table>

**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") returns true()
- geolocation-within-polygon("33 -22", "58 -32", "-78 -55", "48°51.29.6"N 24°17'40.2"E") returns true()

**Geolocation input string formats:**

The geolocation input string must contain latitude and longitude (in that order) separated by whitespace. Each can be in any of the following formats. Combinations are allowed. So latitude can be in one format and longitude can be in another. Latitude values range from +90 to -90 (N to S).
Longitude values range from +180 to -180 (E to W).

**Note:** If single quotes or double quotes are used to delimit the input string argument, this will create a mismatch with the single quotes or double quotes that are used, respectively, to indicate minute-values and second-values. In such cases, the quotes that are used for indicating minute-values and second-values must be escaped by doubling them. In the examples in this section, quotes used to delimit the input string are highlighted in yellow ('"') while unit indicators that are escaped are highlighted in blue ('"').

- **Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)**
  \[D°M'S.S'SS"N/S \quad D°M'S.S'SS"W/E\]
  *Example:* \[33°55'11.11"N \quad 22°44'55.25"W\]

- **Degrees, minutes, decimal seconds, with prefixed sign (+/-); the plus sign for (N/E) is optional**
  \[+/-D°M'S.S's" \quad +/-D°M'S.S's"\]
  *Example:* \[+/-33°55'11.11" \quad -22°44'55.25"\]

- **Degrees, decimal minutes, with suffixed orientation (N/S, E/W)**
  \[D°M.MM'M's"N/S \quad D°M.MM'M's"W/E\]
  *Example:* \[33°55.55'N \quad 22°44.44'W\]

- **Degrees, decimal minutes, with prefixed sign (+/-); the plus sign for (N/E) is optional**
  \[+/-D°M.MM'M' \quad +/-D°M.MM'M'\]
  *Example:* \[+33°55.55' \quad -22°44.44'\]

- **Decimal degrees, with suffixed orientation (N/S, E/W)**
  \[D.DD'DN/S \quad D.DD'DW/E\]
  *Example:* \[33.33N \quad 22.22W\]

- **Decimal degrees, with prefixed sign (+/-); the plus sign for (N/S E/W) is optional**
  \[+/-D.DD \quad +/-D.DD\]
  *Example:* \[33.33 \quad -22.22\]

*Examples of format-combinations:*

- \[33.33N \quad -22°44'55.25"\]
- \[33.33 \quad 22°44'55.25"W\]
- \[33.33 \quad 22.45\]

**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).

<table>
<thead>
<tr>
<th>GPSLatitude</th>
<th>GPSLatitudeRef</th>
<th>GPSLongitude</th>
<th>GPSLongitudeRef</th>
<th>Geolocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 51 21.91</td>
<td>S</td>
<td>151 13 11.73</td>
<td>E</td>
<td>33°51'21.91&quot;S 151°13'11.73&quot;E</td>
</tr>
</tbody>
</table>
geolocation-within-rectangle [altova:]

\[
\text{geolocation-within-rectangle}(\text{Geolocation as } \text{xs:string}, \text{RectCorner-1 as } \text{xs:string}, \text{RectCorner-2 as } \text{xs:string}) \text{ as } \text{xs:boolean}
\]

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 (\textit{formats listed below}). If the Geolocation argument is within the rectangle, then the function returns \text{true()}; otherwise it returns \text{false()}. Latitude values range from +90 to −90 (N to S). Longitude values range from +180 to -180 (E to W).

\textbf{Note:} The \texttt{image-exif-data} function and the Exif metadata’s \texttt{@Geolocation} attribute can be used to supply geolocation input strings.

\textbf{Examples}

\begin{itemize}
  \item \texttt{geolocation-within-rectangle("33 -22", "58 -32", ",-48 24")} returns \text{true()}
  \item \texttt{geolocation-within-rectangle("33 -22", "58 -32", ",48 24")} returns \text{false()}
  \item \texttt{geolocation-within-rectangle("33 -22", "58 -32", ",48°51'29.6" S 24°17'40.2" W")} returns \text{true()}
\end{itemize}

\textbf{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).

\textbf{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 ("').

\begin{itemize}
  \item Degrees, minutes, decimal seconds, with suffixed orientation (N/S, E/W)

\[D°M’.SS”N/S \ D°M’.SS”W/E\]

\texttt{Example: 33°55.11.11’N 22°44.55.25’W}

  \item Degrees, minutes, decimal seconds, with prefixed sign (+/-); the plus sign for (N/E) is optional

\[+/-D°M’.SS” +/-D°M’.SS”\]

\texttt{Example: 33°55.11.11’ -22°44.55.25’}

  \item Degrees, decimal minutes, with suffixed orientation (N/S, E/W)

\[D°M.M’N/S \ D°M.M’W/E\]

\texttt{Example: 33°55.55”N 22°44.44”W}

  \item Degrees, decimal minutes, with prefixed sign (+/-); the plus sign for (N/E) is optional

\[+/-D°M.M’ +/-D°M.M’\]

\end{itemize}
Example: +33°55.55' -22°44.44'

- Decimal degrees, with suffixed orientation (N/S, E/W)
  D.DDN/S  D.DDW/E
  Example: 33.33N  22.22W

- Decimal degrees, with prefixed sign (+/-); the plus sign for (N/S E/W) is optional
  +/-D.DD  +/-D.DD
  Example: 33.33  -22.22

Examples of format-combinations:
33.33N  -22°44'55.25"
33.33  22°44'55.25"W
33.33  22.45

Altova Exif Attribute: Geolocation

The Altova XPath/XQuery Engine generates the custom attribute Geolocation from standard Exif metadata tags. Geolocation is a concatenation of four Exif tags: GPSLatitude, GPSLatitudeRef, GPSLongitude, GPSLongitudeRef, with units added (see table below).

<table>
<thead>
<tr>
<th>GPSLatitude</th>
<th>GPSLatitudeRef</th>
<th>GPSLongitude</th>
<th>GPSLongitudeRef</th>
<th>Geolocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 51 21.91</td>
<td>S</td>
<td>151 13 11.73</td>
<td>E</td>
<td>33°51'21.91&quot;S 151°13'11.73&quot;E</td>
</tr>
</tbody>
</table>

27.1.1.3 XPath/XQuery Functions: Image-Related

The following image-related XPath/XQuery extension functions are supported in the current version of MobileTogether Designer.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of [altova:]. For example: add-years-to-date [altova:].
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function without any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: add-years-to-date(xs:date("2014-01-15"), 10).

XPath functions (used in XPath expressions in XSLT): XP1, XP2, XP3.1
XSLT functions (used in XPath expressions in XSLT):

| XSLT1 | XSLT2 | XSLT3 |

XQuery functions (used in XQuery expressions in XQuery):

| XQ1   | XQ3.1 |

- **suggested-image-file-extension [altova:]**

  `suggested-image-file-extension(Base64String as string) as string? XP3.1 XQ3.1`

  Takes the Base64 encoding of an image file as its argument and returns the file extension of the image as recorded in the Base64-encoding of the image. The returned value is a suggestion based on the image type information available in the encoding. If this information is not available, then an empty string is returned. This function is useful if you wish to save a Base64 image as a file and wish to dynamically retrieve an appropriate file extension.

  **Examples**

  - `suggested-image-file-extension(/MyImages/MobilePhone/Image20141130.01)` returns `'jpg'`
  - `suggested-image-file-extension($XML1/Staff/Person/@photo)` returns `''

  In the examples above, the nodes supplied as the argument of the function are assumed to contain a Base64-encoded image. The first example retrieves `jpg` as the file's type and extension. In the second example, the submitted Base64 encoding does not provide usable file extension information.

- **mt-transform-image [altova:]**

  `mt-transform-image(Base64Image as Base64BinaryString, Size as item()+, Rotation as xs:integer, Quality as xs:integer) as Base64BinaryString XP3.1 XQ3.1`

  Takes a Base64-encoded image as its first argument and returns a transformed Base64-encoded image. The second, third, and fourth arguments are the image parameters that are transformed: size, rotation, and quality.

  - **The Size argument** provides three resizing options.

    | (X, Y) | Absolute pixel values. The aspect ratio is not maintained. The order of height and width does not matter since height and width are automatically selected according to the long and short sides of the image. The value is entered as a sequence of two integer items; the parentheses are required.
    | X     | Proportionally resizes the image with `X` as the new longer side in pixels; aspect ratio is maintained. The value is an integer, and is entered without quotes.
    | 'X%'  | Resizes the image to the given percentage of the original dimensions. The value must be entered as a string, in quotes.

  - **Rotation** can be one of the following values: 90, 180, 270, −90, −180, −270. These are rotation values in degrees of a circle. Positive values rotate the image clockwise; negative values rotate the image counter-clockwise. Note that you can use the Altova Exif attribute `OrientationDegree` to obtain the current rotation of the image in degrees (0, 90, 180, 270) from the Exif `Orientation` tag of the image. However, since the `OrientationDegree` attribute is obtained from the `Orientation` tag of the Exif data, it will be available only if the `Orientation` tag is present in the Exif data (see the description of `OrientationDegree` below).
Quality can be any value between 0 and 100 and refers to values on the IJG quality scale for JPEG compression; it is not a percentage indicator of quality. The tradeoff is between file size and quality. For a full-color source image, 75 is generally considered an optimal value. If 75 produces unsatisfactory results, increase the value.

Note: If Exif data is present in the original image, it will be removed during the transformation, and the transformed image will not contain Exif data.

Examples

- **mt-transform-image**(Images/Image[@id='43'], '50%', 90, 75)
  The function takes as its input an image that is stored as a Base64-encoded string in the descendant Images/Image node that has an @id value of 43. The function returns a transformed image. The transformed image is resized to 50%, rotated 90 degrees clockwise, and given a quality level of 75.

- **mt-transform-image**(Images/Image[@id='43'], 400, 90, 75)
  The function produces the same result as the previous example, except that the long side is set to a specific value of 400 pixels; the aspect ratio of the original image is maintained.

- **mt-transform-image**(Images/Image[@id='43'], (400, 280), image-exif-data($XML1/$XML1/Images/ReferenceImage)/@OrientationDegree, 75)
  This example selects the same image as in the previous examples, and sets the same quality value (75). The image size is set at 400x280 pixels, and the Rotation value is obtained from the @OrientationDegree attribute of a Base64-encoded image in the ReferenceImage node.

Altova Exif Attribute: OrientationDegree

The Altova XPath/XQuery Engine generates the custom attribute OrientationDegree from the Exif metadata tag Orientation. OrientationDegree translates the standard Exif tag Orientation from an integer value (1, 8, 3, or 6) to the respective degree values of each (0, 90, 180, 270), as shown in the figure below. Note that there are no translations of the Orientation values of 2, 4, 5, 7. (These orientations are obtained by flipping image 1 across its vertical center axis to get the image with a value of 2, and then rotating this image in 90-degree jumps clockwise to get the values of 7, 4, and 5, respectively).
Listing of standard Exif meta tags

- ImageWidth
- ImageLength
- BitsPerSample
- Compression
- PhotometricInterpretation
- Orientation
- SamplesPerPixel
- PlanarConfiguration
- YCbCrSubSampling
- YCbCrPositioning
- XResolution
- YResolution
- ResolutionUnit
- StripOffsets
- RowsPerStrip
- StripByteCounts
- JPEGInterchangeFormat
- JPEGInterchangeFormatLength
- TransferFunction
- WhitePoint
- PrimaryChromaticities
- YCbCrCoefficients
- ReferenceBlackWhite
- DateTime
- ImageDescription
- Make
• Model
• Software
• Artist
• Copyright

-----------------------------

• ExifVersion
• FlashpixVersion
• ColorSpace
• ComponentsConfiguration
• CompressedBitsPerPixel
• PixelXDimension
• PixelYDimension
• MakerNote
• UserComment
• RelatedSoundFile
• DateTimeOriginal
• DateTimeDigitized
• SubSecTime
• SubSecTimeOriginal
• SubSecTimeDigitized
• ExposureTime
• FNumber
• ExposureProgram
• SpectralSensitivity
• ISOSpeedRatings
• OECF
• ShutterSpeedValue
• ApertureValue
• BrightnessValue
• ExposureBiasValue
• MaxApertureValue
• SubjectDistance
• MeteringMode
• LightSource
• Flash
• FocalLength
• SubjectArea
• FlashEnergy
• SpatialFrequencyResponse
• FocalPlaneXResolution
• FocalPlaneYResolution
• FocalPlaneResolutionUnit
• SubjectLocation
• ExposureIndex
• SensingMethod
• FileSource
• SceneType
• CFAPattern
• CustomRendered
• ExposureMode
• WhiteBalance
• DigitalZoomRatio
• FocalLengthIn35mmFilm
• SceneCaptureType
• GainControl
• Contrast
• Saturation
• Sharpness
• DeviceSettingDescription
• SubjectDistanceRange
• ImageUniqueID

-------------------------------

• GPSVersionID
• GPSLatitudeRef
• GPSLatitude
• GPSLongitudeRef
• GPSLongitude
• GPSAltitudeRef
• GPSAltitude
• GPSTimeStamp
• GPSSatellites
• GPSStatus
• GPSMeasureMode
• GPSDOP
• GPSSpeedRef
• GPSSpeed
• GPSTrackRef
• GPSTrack
• GPSImgDirectionRef
• GPSImgDirection
• GPSMapDatum
• GPSDestLatitudeRef
• GPSDestLatitude
• GPSDestLongitudeRef
• GPSDestLongitude
• GPSDestBearingRef
• GPSDestBearing
• GPSDestDistanceRef
• GPSDestDistance
• GPSProcessingMethod
• GPSAreaInformation
• GPSDateStamp
• GPSDifferential

▼ image-exif-data [altova:]

image-exif-data(Base64BinaryString as string) as element?  XP3.1 XQ3.1

Takes a Base64-encoded JPEG image as its argument and returns an element called Exif that contains the Exif metadata of the image. The Exif metadata is created as attribute-value pairs of the Exif element. The attribute names are the Exif data tags found in the Base64 encoding. The list of Exif-specification tags is given below. If a vendor-specific tag is present in the Exif data, this tag and its value will also be returned as an attribute-value pair. Additional to the standard Exif metadata tags (see list below), Altova-specific attribute-value pairs are also generated. These Altova Exif attributes are listed below.

Examples
• To access any one attribute, use the function like this:
  \texttt{image-exif-data}(//MyImages/Image20141130.01)/@GPSLatitude
  \texttt{image-exif-data}(//MyImages/Image20141130.01)/@Geolocation

• To access all the attributes, use the function like this:
  \texttt{image-exif-data}(//MyImages/Image20141130.01)/@*

• To access the names of all the attributes, use the following expression:
  \texttt{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.

\textbf{Altova Exif Attribute: Geolocation}

The Altova XPath/XQuery Engine generates the custom attribute \texttt{Geolocation} from standard Exif metadata tags. \texttt{Geolocation} is a concatenation of four Exif tags: \texttt{GPSLatitude}, \texttt{GPSLatitudeRef}, \texttt{GPSLongitude}, \texttt{GPSLongitudeRef}, with units added (see table below).

<table>
<thead>
<tr>
<th>GPSLatitude</th>
<th>GPSLatitudeRef</th>
<th>GPSLongitude</th>
<th>GPSLongitudeRef</th>
<th>Geolocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>33 51 21.91</td>
<td>S</td>
<td>151 13 11.73</td>
<td>E</td>
<td>33°51'21.91&quot;S 151°13'11.73&quot;E</td>
</tr>
</tbody>
</table>

\textbf{Altova Exif Attribute: OrientationDegree}

The Altova XPath/XQuery Engine generates the custom attribute \texttt{OrientationDegree} from the Exif metadata tag \texttt{Orientation}.

\texttt{OrientationDegree} translates the standard Exif tag \texttt{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 integer values (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 \texttt{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
27.1.1.4 XPath/XQuery Functions: Numeric

Altova's numeric extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to
the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of [altova:]. For example: \texttt{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: \texttt{add-years-to-date(xs:date("2014-01-15"), 10)}.

### Auto-numbering functions

\textbf{generate-auto-number [altova:]} 

\begin{verbatim}
  generate-auto-number (ID as xs:string, StartsWith as xs:double, Increment as xs:double,
  ResetOnChange as xs:string) as xs:integer
\end{verbatim}

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 \texttt{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 \texttt{Increment} argument. In effect, the generate-auto-number function creates a counter having a name specified by the \texttt{ID} argument, with this counter being incremented each time the function is called. If the value of the \texttt{ResetOnChange} argument changes from that of the previous function call, then the value of the number to be generated is reset to the \texttt{StartsWith} value. Auto-numbering can also be reset by using the \texttt{reset-auto-number} function.

\textbf{Examples}

- \texttt{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 \texttt{reset-auto-number} function, like this: \texttt{reset-auto-number ("ChapterNumber")}.

\textbf{reset-auto-number [altova:]} 

\begin{verbatim}
  reset-auto-number (ID as xs:string) as xs:integer
\end{verbatim}

This function resets the number of the auto-numbering counter named in the \texttt{ID} argument. The number is reset to the number specified by the \texttt{StartsWith} argument of the generate-auto-number function that created the counter named in the \texttt{ID} argument.

\textbf{Examples}

- \texttt{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 \texttt{StartsWith} argument of the generate-auto-number function that created ChapterNumber.
Numeric functions

hex-string-to-integer [altova:]

\[ \text{hex-string-to-integer(HexString as xs:string) as xs:integer} \]

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**
- \( \text{hex-string-to-integer('1')} \) returns 1
- \( \text{hex-string-to-integer('9')} \) returns 9
- \( \text{hex-string-to-integer('A')} \) returns 10
- \( \text{hex-string-to-integer('B')} \) returns 11
- \( \text{hex-string-to-integer('C')} \) returns 12
- \( \text{hex-string-to-integer('F')} \) returns 15
- \( \text{hex-string-to-integer('G')} \) returns an error
- \( \text{hex-string-to-integer('10')} \) returns 16
- \( \text{hex-string-to-integer('01')} \) returns 1
- \( \text{hex-string-to-integer('20')} \) returns 32
- \( \text{hex-string-to-integer('21')} \) returns 33
- \( \text{hex-string-to-integer('5A')} \) returns 90
- \( \text{hex-string-to-integer('USA')} \) returns an error

integer-to-hex-string [altova:]

\[ \text{integer-to-hex-string(Integer as xs:integer) as xs:string} \]

Takes an integer argument and returns its Base-16 equivalent as a string.

**Examples**
- \( \text{integer-to-hex-string(1)} \) returns '1'
- \( \text{integer-to-hex-string(9)} \) returns '9'
- \( \text{integer-to-hex-string(10)} \) returns 'A'
- \( \text{integer-to-hex-string(11)} \) returns 'B'
- \( \text{integer-to-hex-string(15)} \) returns 'F'
- \( \text{integer-to-hex-string(16)} \) returns '10'
- \( \text{integer-to-hex-string(32)} \) returns '20'
- \( \text{integer-to-hex-string(33)} \) returns '21'
- \( \text{integer-to-hex-string(90)} \) returns '5A'

Number-formatting functions

mt-format-number [altova:]

\[ \text{mt-format-number(Number as xs:numeric, PictureString as xs:string) as xs:string} \]

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

### 27.1.1.5 XPath/XQuery Functions: Schema

The Altova extension functions listed below return schema information. Given below are descriptions of the functions, together with (i) examples and (ii) a listing of schema components and their respective properties.

**Schema information from schema documents**

The function `altova:schema` has two arguments: one with zero arguments and the other with two arguments. The zero-argument function returns the whole schema. You can then, from this starting point, navigate into the schema to locate the schema components you want. The two-argument function returns a specific component kind that is identified by its QName. In both cases, the return value is a function. To navigate into the returned component, you must select a property of that specific component. If the property is a non-atomic item (that is, if it is a component), then you can navigate further by selecting a property of this component. If the selected property is an atomic item, then the value of the item is returned and you cannot navigate any further.

**Note:** In XQuery expressions, the schema must be explicitly imported. In XPath expressions, the schema must have been imported into the processing environment, for example, into XSLT with the `xslt:import` instruction.
Schema information from XML nodes
The function `altova:type` submits the node of an XML document and returns the node's type information from the PSVI.

Note about naming of functions and language applicability
Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.
- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of `[altova:]`. For example: `add-years-to-date [altova:]`.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function without any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

<table>
<thead>
<tr>
<th>XPath functions (used in XPath expressions in XSLT)</th>
<th>X1</th>
<th>X2</th>
<th>X3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSLT functions (used in XPath expressions in XSLT)</td>
<td>XSLT1</td>
<td>XSLT2</td>
<td>XSLT3</td>
</tr>
<tr>
<td>XQuery functions (used in XQuery expressions in XQuery)</td>
<td>XQ1</td>
<td>XQ3.1</td>
<td></td>
</tr>
</tbody>
</table>

**Schema (zero arguments)**

`altova:schema()` as (function(xs:string) as item()*)? XP3.1 XQ3.1
Returns the `schema` component as a whole. You can navigate further into the `schema` component by selecting one of the `schema` component's properties.

- If this property is a component, you can navigate another step deeper by selecting one of this component's properties. This step can be repeated to navigate further into the schema.
- If the component is an atomic value, the atomic value is returned and you cannot navigate any deeper.

The properties of the `schema` component are:

"type definitions"
"attribute declarations"
"element declarations"
"attribute group definitions"
"model group definitions"
"notation declarations"
"identity-constraint definitions"

The properties of all other component kinds (besides `schema`) are listed below.

**Note:** In XQuery expressions, the schema must be explicitly imported. In XPath expressions, the schema must have been imported into the processing environment, for example, into XSLT with the `xslt:import` instruction.

**Examples**
- `import schema "" at "C:\Test\ExpReport.xsd"; for $typedef in altova:schema()`
  ("type definitions")
return $typedef ("name") returns the names of all simple types or complex types in the schema

- import schema "" at "C:\Test\ExpReport.xsd";
  altova:schema () ("type definitions")[1] ("name") returns the name of the first of all simple types or complex types in the schema

**Components and their properties**

- **Assertion**
  
<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Assertion&quot;</td>
</tr>
<tr>
<td>test</td>
<td>XPath Property Record</td>
<td></td>
</tr>
</tbody>
</table>

- **Attribute Declaration**
  
<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Declaration&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the attribute</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the attribute</td>
</tr>
<tr>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
<tr>
<td>scope</td>
<td>A function with properties</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(&quot;class&quot;: &quot;Scope&quot;, &quot;variety&quot;: &quot;global&quot; or &quot;local&quot;, &quot;parent&quot;: the containing Complex Type or Attribute Group)</td>
<td></td>
</tr>
<tr>
<td>value constraint</td>
<td>If present, a function with properties</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(&quot;class&quot;: &quot;Value Constraint&quot;, &quot;variety&quot;: &quot;fixed&quot; or &quot;default&quot;, &quot;value&quot;: atomic value, &quot;lexical form&quot;: string. Note that the &quot;value&quot; property is not available for namespace-sensitive types</td>
<td></td>
</tr>
<tr>
<td>inheritable</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

- **Attribute Group Declaration**
  
<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Group Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the attribute group</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the attribute group</td>
</tr>
<tr>
<td>attribute uses</td>
<td>Sequence of (Attribute Use)</td>
<td></td>
</tr>
<tr>
<td>attribute wildcard</td>
<td>Optional Attribute Wildcard</td>
<td></td>
</tr>
</tbody>
</table>

© 2015-2021 Altova GmbH
### Attribute Use

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Use&quot;</td>
</tr>
<tr>
<td>required</td>
<td>boolean</td>
<td>true if the attribute is required, false if optional</td>
</tr>
<tr>
<td>value constraint</td>
<td></td>
<td>See Attribute Declaration</td>
</tr>
<tr>
<td>inheritable</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

### Attribute Wildcard

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Wildcard&quot;</td>
</tr>
<tr>
<td>namespace constraint</td>
<td>function</td>
<td>properties (&quot;class&quot;: &quot;Namespace Constraint&quot;, &quot;variety&quot;: &quot;any&quot;</td>
</tr>
<tr>
<td>process contents</td>
<td>string</td>
<td>(&quot;strict&quot;</td>
</tr>
</tbody>
</table>

### Complex Type

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Complex Type&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>base type definition</td>
<td>Complex Type</td>
<td>Definition</td>
</tr>
<tr>
<td>final</td>
<td>Sequence of strings</td>
<td>(&quot;restriction&quot;</td>
</tr>
<tr>
<td>context</td>
<td>Empty sequence</td>
<td>(not implemented)</td>
</tr>
<tr>
<td>derivation method</td>
<td>string</td>
<td>(&quot;restriction&quot;</td>
</tr>
<tr>
<td>abstract</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>attribute uses</td>
<td>Sequence of Attribute Use</td>
<td></td>
</tr>
<tr>
<td>attribute wildcard</td>
<td>Optional Attribute Wildcard</td>
<td></td>
</tr>
<tr>
<td>content type</td>
<td>function</td>
<td>properties: (&quot;class&quot;:&quot;Content Type&quot;, &quot;variety&quot;:string (&quot;element-only&quot;</td>
</tr>
</tbody>
</table>
### Optional Particle, "open content":
function with properties ("class":"Open Content", "mode": string ("interleave"|"suffix"), "wildcard": Wildcard), "simple type definition": Simple Type)

| prohibited substitutions | Sequence of strings ("restriction"|"extension") |
|--------------------------|-----------------------------------------------|
| assertions               | Sequence of Assertion                          |

#### Element Declaration

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Complex Type&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
<tr>
<td>type table</td>
<td>function with properties (&quot;class&quot;:&quot;Type Table&quot;, &quot;alternatives&quot;: sequence of Type Alternative, &quot;default type definition&quot;: Simple Type or Complex Type)</td>
<td></td>
</tr>
<tr>
<td>scope</td>
<td>function with properties (&quot;class&quot;:&quot;Scope&quot;, &quot;variety&quot;: (&quot;global&quot;</td>
<td>&quot;local&quot;), &quot;parent&quot;: optional Complex Type)</td>
</tr>
<tr>
<td>value constraint</td>
<td>see Attribute Declaration</td>
<td></td>
</tr>
<tr>
<td>nillable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>identity-constraint</td>
<td>Sequence of Identity Constraint</td>
<td></td>
</tr>
<tr>
<td>definitions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>substitution group</td>
<td>Sequence of Element Declaration</td>
<td></td>
</tr>
<tr>
<td>affiliations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>substitution group</td>
<td>Sequence of strings (&quot;restriction&quot;</td>
<td>&quot;extension&quot;)</td>
</tr>
<tr>
<td>exclusions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>disallowed substitutions</td>
<td>Sequence of strings (&quot;restriction&quot;</td>
<td>&quot;extension&quot;</td>
</tr>
<tr>
<td>abstract</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

#### Element Wildcard

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Wildcard&quot;</td>
</tr>
<tr>
<td>namespace constraint</td>
<td>function with properties (&quot;class&quot;:</td>
<td></td>
</tr>
</tbody>
</table>
### XSLT and XPath/XQuery Functions

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>The name of the facet, for example &quot;minLength&quot; or &quot;enumeration&quot;</td>
</tr>
<tr>
<td>value</td>
<td>depends on facet</td>
<td>The value of the facet</td>
</tr>
<tr>
<td>fixed</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>typed-value</td>
<td>array(xs:anyAtomicType*)</td>
<td>An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the &quot;value&quot; property is a sequence of strings, regardless of the actual type)</td>
</tr>
</tbody>
</table>

### Identity Constraint

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Identity-Constraint Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the constraint</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the constraint</td>
</tr>
<tr>
<td>identity-constraint category</td>
<td>string (&quot;key&quot;</td>
<td>&quot;unique&quot;</td>
</tr>
<tr>
<td>selector</td>
<td>XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>fields</td>
<td>Sequence of XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>referenced key</td>
<td>(For keyRef only): Identity Constraint</td>
<td>The corresponding key constraint</td>
</tr>
</tbody>
</table>

### Model Group

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Model Group&quot;</td>
</tr>
<tr>
<td>compositor</td>
<td>string (&quot;sequence&quot;</td>
<td>&quot;choice&quot;</td>
</tr>
<tr>
<td>particles</td>
<td>Sequence of Particle</td>
<td></td>
</tr>
</tbody>
</table>
### Model Group Definition

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Model Group Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the model group</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the model group</td>
</tr>
<tr>
<td>model group</td>
<td>Model Group</td>
<td></td>
</tr>
</tbody>
</table>

### Notation

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Notation Declaration&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the notation</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the notation</td>
</tr>
<tr>
<td>system identifier</td>
<td>anyURI</td>
<td></td>
</tr>
<tr>
<td>public identifier</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

### Particle

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Particle&quot;</td>
</tr>
<tr>
<td>min occurs</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>max occurs</td>
<td>integer, or string(&quot;unbounded&quot;)</td>
<td></td>
</tr>
<tr>
<td>term</td>
<td>Element Declaration, Element Wildcard, or ModelGroup</td>
<td></td>
</tr>
</tbody>
</table>

### Simple Type

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Simple Type Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>final</td>
<td>Sequence of string(&quot;restriction&quot;</td>
<td>&quot;extension&quot;</td>
</tr>
<tr>
<td>context</td>
<td>containing component</td>
<td></td>
</tr>
<tr>
<td>base type definition</td>
<td>Simple Type</td>
<td></td>
</tr>
<tr>
<td>facets</td>
<td>Sequence of Facet</td>
<td></td>
</tr>
<tr>
<td>fundamental facets</td>
<td>Empty sequence (not implemented)</td>
<td></td>
</tr>
<tr>
<td>variety</td>
<td>string (&quot;atomic&quot;</td>
<td>&quot;list&quot;</td>
</tr>
<tr>
<td>----------------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>primitive type definition</td>
<td>Simple Type</td>
<td></td>
</tr>
<tr>
<td>item type definition</td>
<td>(for list types only) Simple Type</td>
<td></td>
</tr>
<tr>
<td>member type definitions</td>
<td>(for union types only) Sequence of Simple Type</td>
<td></td>
</tr>
</tbody>
</table>

Type Alternative

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Type Alternative&quot;</td>
</tr>
<tr>
<td>test</td>
<td>XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
</tbody>
</table>

XPath Property Record

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace bindings</td>
<td>Sequence of functions with properties (&quot;prefix&quot;: string, &quot;namespace&quot;: anyURI)</td>
<td></td>
</tr>
<tr>
<td>default namespace</td>
<td>anyURI</td>
<td></td>
</tr>
<tr>
<td>base URI</td>
<td>anyURI</td>
<td>The static base URI of the XPath expression</td>
</tr>
<tr>
<td>expression</td>
<td>string</td>
<td>The XPath expression as a string</td>
</tr>
</tbody>
</table>

Effects (two arguments)

```xml
altova:schema(ComponentKind as xs:string, Name as xs:QName) as (function(xs:string) as item()*)? XP3.1 XQ3.1
```

Returns the component kind that is specified in the first argument which has a name that is the same as the name supplied in the second argument. You can navigate further by selecting one of the component's properties.

- If this property is a component, you can navigate another step deeper by selecting one of this component's properties. This step can be repeated to navigate further into the schema.
- If the property is an atomic value, the atomic value is returned and you cannot navigate any deeper.

Note: In XQuery expressions, the schema must be explicitly imported. In XPath expressions, the schema must have been imported into the processing environment, for example, into XSLT with the `xslt:import` instruction.

Examples

- import schema "" at "C:\Test\ExpReport.xsd";
- altova:schema("element declaration", xs:QName("OrgChart"))("type definition")
returns the kind property of the term of the third particles component. This particles component is a descendant of the element declaration having a QName of OrgChart

```
import schema "" at "C:\Test\ExpReport.xsd";
let $typedef := altova:schema("type definition", xs:QName("emailType"))
for $facet in $typedef ("facets")
return [$facet ("kind"), $facet ("value")]
returns, for each facet of each emailType component, an array containing that facet's kind and value

Components and their properties

 Assertion

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Assertion&quot;</td>
</tr>
<tr>
<td>test</td>
<td>XPath Property Record</td>
<td></td>
</tr>
</tbody>
</table>

 Attribute Declaration

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Declaration&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the attribute</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the attribute</td>
</tr>
<tr>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
<tr>
<td>scope</td>
<td>A function with properties</td>
<td></td>
</tr>
<tr>
<td>value constraint</td>
<td>If present, a function with properties</td>
<td></td>
</tr>
<tr>
<td>inheritable</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

 Attribute Group Declaration

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Group Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the attribute group</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the attribute group</td>
</tr>
</tbody>
</table>
### Attribute Use

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Use&quot;</td>
</tr>
<tr>
<td>required</td>
<td>boolean</td>
<td>true if the attribute is required, false if optional</td>
</tr>
<tr>
<td>value constraint</td>
<td></td>
<td>See Attribute Declaration</td>
</tr>
<tr>
<td>inheritable</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

### Attribute Wildcard

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Wildcard&quot;</td>
</tr>
<tr>
<td>namespace constraint</td>
<td>function with properties (&quot;class&quot;: &quot;Namespace Constraint&quot;, &quot;variety&quot;: &quot;any&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;enumeration&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;not&quot;,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;namespaces&quot;:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>sequence of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xs:.anyURI,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&quot;disallowed names&quot;:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>list containing QNames and/or the strings &quot;defined&quot; and &quot;definedSiblings&quot;</td>
<td></td>
</tr>
<tr>
<td>process contents</td>
<td>string</td>
<td>(&quot;strict&quot;</td>
</tr>
</tbody>
</table>

### Complex Type

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Complex Type&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>base type definition</td>
<td>Complex Type Definition</td>
<td></td>
</tr>
<tr>
<td>final</td>
<td>Sequence of strings</td>
<td>(&quot;restriction&quot;</td>
</tr>
<tr>
<td>context</td>
<td>Empty sequence (not implemented)</td>
<td></td>
</tr>
<tr>
<td>derivation method</td>
<td>string</td>
<td>(&quot;restriction&quot;</td>
</tr>
<tr>
<td>abstract</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>attribute uses</td>
<td>Sequence of Attribute Use</td>
<td></td>
</tr>
<tr>
<td>attribute wildcard</td>
<td>Optional Attribute Wildcard</td>
<td></td>
</tr>
</tbody>
</table>
### Element Declaration

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Complex Type&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
<tr>
<td>type table</td>
<td>function with properties (&quot;class&quot;:&quot;Type Table&quot;, &quot;alternatives&quot;: sequence of Type Alternative, &quot;default type definition&quot;: Simple Type or Complex Type)</td>
<td></td>
</tr>
<tr>
<td>scope</td>
<td>function with properties (&quot;class&quot;:&quot;Scope&quot;, &quot;variety&quot;: (&quot;global&quot;</td>
<td>&quot;local&quot;), &quot;parent&quot;: optional Complex Type)</td>
</tr>
<tr>
<td>value constraint</td>
<td>see Attribute Declaration</td>
<td></td>
</tr>
<tr>
<td>nillable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>identity-constraint definitions</td>
<td>Sequence of Identity Constraint</td>
<td></td>
</tr>
<tr>
<td>substitution group affiliations</td>
<td>Sequence of Element Declaration</td>
<td></td>
</tr>
<tr>
<td>substitution group exclusions</td>
<td>Sequence of strings (&quot;restriction&quot;</td>
<td>&quot;extension&quot;)</td>
</tr>
<tr>
<td>disallowed substitutions</td>
<td>Sequence of strings (&quot;restriction&quot;</td>
<td>&quot;extension&quot;</td>
</tr>
<tr>
<td>abstract</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>
### XSLT and XPath/XQuery Functions

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Wildcard&quot;</td>
</tr>
<tr>
<td>namespace constraint</td>
<td>function with properties (&quot;class&quot;: &quot;Namespace Constraint&quot;, &quot;variety&quot;: &quot;any&quot;</td>
<td>&quot;enumeration&quot;</td>
</tr>
<tr>
<td>process contents</td>
<td>string</td>
<td>(&quot;strict&quot;</td>
</tr>
</tbody>
</table>

#### Facet

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>The name of the facet, for example &quot;minLength&quot; or &quot;enumeration&quot;</td>
</tr>
<tr>
<td>value</td>
<td>depends on facet</td>
<td>The value of the facet</td>
</tr>
<tr>
<td>fixed</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>typed-value</td>
<td>For the enumeration facet only, array(xs:anyAtomicType*)</td>
<td>An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the &quot;value&quot; property is a sequence of strings, regardless of the actual type)</td>
</tr>
</tbody>
</table>

#### Identity Constraint

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Identity-Constraint Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the constraint</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the constraint</td>
</tr>
<tr>
<td>identity-constraint</td>
<td>string (&quot;key&quot;</td>
<td>&quot;unique&quot;</td>
</tr>
<tr>
<td>selector</td>
<td>XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>fields</td>
<td>Sequence of XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>referenced key</td>
<td>(For keyRef only): Identity Constraint</td>
<td>The corresponding key constraint</td>
</tr>
</tbody>
</table>

#### Model Group

| Property name     | Property type | Property value |
| compositors | string ("sequence"|"choice"|"all") |
|-------------|---------------------------------|
| particles   | Sequence of Particle             |

- **Model Group Definition**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Model Group Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the model group</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the model group</td>
</tr>
<tr>
<td>model group</td>
<td>Model Group</td>
<td></td>
</tr>
</tbody>
</table>

- **Notation**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Notation Declaration&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the notation</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the notation</td>
</tr>
<tr>
<td>system identifier</td>
<td>anyURI</td>
<td></td>
</tr>
<tr>
<td>public identifier</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

- **Particle**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Particle&quot;</td>
</tr>
<tr>
<td>min occurs</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>max occurs</td>
<td>integer, or string(&quot;unbounded&quot;)</td>
<td></td>
</tr>
<tr>
<td>term</td>
<td>Element Declaration, Element Wildcard, or ModelGroup</td>
<td></td>
</tr>
</tbody>
</table>

- **Simple Type**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Simple Type Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>final</td>
<td>Sequence of string(&quot;restriction&quot;</td>
<td>&quot;extension&quot;</td>
</tr>
<tr>
<td>context</td>
<td>containing component</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>----------------------</td>
<td></td>
</tr>
<tr>
<td>base type definition</td>
<td>Simple Type</td>
<td></td>
</tr>
<tr>
<td>facets</td>
<td>Sequence of Facet</td>
<td></td>
</tr>
<tr>
<td>fundamental facets</td>
<td>Empty sequence (not implemented)</td>
<td></td>
</tr>
<tr>
<td>variety</td>
<td>string (&quot;atomic&quot;,&quot;list&quot;,&quot;union&quot;)</td>
<td></td>
</tr>
<tr>
<td>primitive type definition</td>
<td>Simple Type</td>
<td></td>
</tr>
<tr>
<td>item type definition</td>
<td>(for list types only) Simple Type</td>
<td></td>
</tr>
<tr>
<td>member type definitions</td>
<td>(for union types only) Sequence of Simple Type</td>
<td></td>
</tr>
</tbody>
</table>

- **Type Alternative**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Type Alternative&quot;</td>
</tr>
<tr>
<td>test</td>
<td>XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
</tbody>
</table>

- **XPath Property Record**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace bindings</td>
<td>Sequence of functions with properties (&quot;prefix&quot;: string, &quot;namespace&quot;: anyURI)</td>
<td></td>
</tr>
<tr>
<td>default namespace</td>
<td>anyURI</td>
<td></td>
</tr>
<tr>
<td>base URI</td>
<td>anyURI</td>
<td>The static base URI of the XPath expression</td>
</tr>
<tr>
<td>expression</td>
<td>string</td>
<td>The XPath expression as a string</td>
</tr>
</tbody>
</table>

- **Type**

```xml
altova:type(Node as item?) as (function(xs:string) as item()*)? XQ3.1 XQ3.1
```

The function `altova:type` submits an element or attribute node of an XML document and returns the node's type information from the PSVI.

**Note:** The XML document must have a schema declaration so that the schema can be referenced.

**Examples**

- ```xml
    for $element in //Email
    let $type := altova:type($element)
    return $type
  returns a function that contains the node's type information
  ```

- ```xml
    for $element in //Email
  ```
let $type := altova:type($element)
return $type ("kind")
takes the node's type component (Simple Type or Complex Type) and returns the value of the component's kind property

Components and their properties

Assertion

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Assertion&quot;</td>
</tr>
<tr>
<td>test</td>
<td>XPath Property Record</td>
<td></td>
</tr>
</tbody>
</table>

Attribute Declaration

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Declaration&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the attribute</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the attribute</td>
</tr>
<tr>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
<tr>
<td>scope</td>
<td>A function with properties</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(&quot;class&quot;: &quot;Scope&quot;, &quot;variety&quot;: &quot;global&quot; or &quot;local&quot;, &quot;parent&quot;: the containing Complex Type or Attribute Group)</td>
<td></td>
</tr>
<tr>
<td>value constraint</td>
<td>If present, a function with properties (&quot;class&quot;: &quot;Value Constraint&quot;, &quot;variety&quot;: &quot;fixed&quot; or &quot;default&quot;, &quot;value&quot;: atomic value, &quot;lexical form&quot;: string. Note that the &quot;value&quot; property is not available for namespace-sensitive types</td>
<td></td>
</tr>
<tr>
<td>inheritable</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

Attribute Group Declaration

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Group Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the attribute group</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the attribute group</td>
</tr>
<tr>
<td>attribute uses</td>
<td>Sequence of (Attribute Use)</td>
<td></td>
</tr>
<tr>
<td>attribute wildcard</td>
<td>Optional Attribute Wildcard</td>
<td></td>
</tr>
</tbody>
</table>

Attribute Use
### XSLT and XPath/XQuery Functions

#### Appendixes

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Attribute Use&quot;</td>
</tr>
<tr>
<td>required</td>
<td>boolean</td>
<td>true if the attribute is required, false if optional</td>
</tr>
<tr>
<td>value constraint</td>
<td>See Attribute Declaration</td>
<td></td>
</tr>
<tr>
<td>inheritable</td>
<td>boolean</td>
<td></td>
</tr>
</tbody>
</table>

**Attribute Wildcard**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Wildcard&quot;</td>
</tr>
<tr>
<td>namespace constraint</td>
<td>function with properties (&quot;class&quot;:&quot;Namespace Constraint&quot;, &quot;variety&quot;:&quot;any&quot;</td>
<td>enumeration&quot;</td>
</tr>
<tr>
<td>process contents</td>
<td>string (&quot;strict&quot;</td>
<td>&quot;lax&quot;</td>
</tr>
</tbody>
</table>

**Complex Type**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Complex Type&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>base type definition</td>
<td>Complex Type Definition</td>
<td></td>
</tr>
<tr>
<td>final</td>
<td>Sequence of strings (&quot;restriction&quot;</td>
<td>&quot;extension&quot;)</td>
</tr>
<tr>
<td>context</td>
<td>Empty sequence (not implemented)</td>
<td></td>
</tr>
<tr>
<td>derivation method</td>
<td>string (&quot;restriction&quot;</td>
<td>&quot;extension&quot;)</td>
</tr>
<tr>
<td>abstract</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>attribute uses</td>
<td>Sequence of Attribute Use</td>
<td></td>
</tr>
<tr>
<td>attribute wildcard</td>
<td>Optional Attribute Wildcard</td>
<td></td>
</tr>
<tr>
<td>content type</td>
<td>function with properties: (&quot;class&quot;:&quot;Content Type&quot;, &quot;variety&quot;:string (&quot;element-only&quot;</td>
<td>&quot;empty&quot;</td>
</tr>
</tbody>
</table>

© 2015-2021 Altova GmbH

Altova MobileTogether Designer
<table>
<thead>
<tr>
<th></th>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element Declaration</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kind</td>
<td>string</td>
<td>&quot;Complex Type&quot;</td>
</tr>
<tr>
<td></td>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td></td>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td></td>
<td>type definition</td>
<td>Simple Type or Complex Type</td>
<td></td>
</tr>
<tr>
<td></td>
<td>type table</td>
<td>function with properties</td>
<td>(&quot;class&quot;:&quot;Type Table&quot;,&quot;alternatives&quot;:sequence of Type Alternative,&quot;default type definition&quot;:Simple Type or Complex Type)</td>
</tr>
<tr>
<td></td>
<td>scope</td>
<td>function with properties</td>
<td>(&quot;class&quot;:&quot;Scope&quot;,&quot;variety&quot;:(&quot;global&quot;,&quot;local&quot;),&quot;parent&quot;:optional Complex Type)</td>
</tr>
<tr>
<td></td>
<td>value constraint</td>
<td>see Attribute Declaration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>nillable</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>identity-constraint definitions</td>
<td>Sequence of Identity Constraint</td>
<td></td>
</tr>
<tr>
<td></td>
<td>substitution group affiliations</td>
<td>Sequence of Element Declaration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>substitution group exclusions</td>
<td>Sequence of strings</td>
<td>(&quot;restriction&quot;,&quot;extension&quot;)</td>
</tr>
<tr>
<td></td>
<td>disallowed substitutions</td>
<td>Sequence of strings</td>
<td>(&quot;restriction&quot;,&quot;extension&quot;,&quot;substitution&quot;)</td>
</tr>
<tr>
<td></td>
<td>abstract</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td><strong>Element Wildcard</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>kind</td>
<td>string</td>
<td>&quot;Wildcard&quot;</td>
</tr>
<tr>
<td></td>
<td>namespace constraint</td>
<td>function with properties</td>
<td>(&quot;class&quot;:&quot;Namespace Constraint&quot;,&quot;variety&quot;:&quot;any&quot;,&quot;enumeration&quot;,&quot;not&quot;),</td>
</tr>
</tbody>
</table>
### Facet

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>The name of the facet, for example &quot;minLength&quot; or &quot;enumeration&quot;</td>
</tr>
<tr>
<td>value</td>
<td>depends on facet</td>
<td>The value of the facet</td>
</tr>
<tr>
<td>fixed</td>
<td>boolean</td>
<td></td>
</tr>
<tr>
<td>typed-value</td>
<td>array(xs:anyAtomicType*)</td>
<td>An array containing the enumeration values, each of which may in general be a sequence of atomic values. (Note: for the enumeration facet, the &quot;value&quot; property is a sequence of strings, regardless of the actual type)</td>
</tr>
</tbody>
</table>

### Identity Constraint

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Identity-Constraint Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the constraint</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the constraint</td>
</tr>
<tr>
<td>identity-constraint</td>
<td>string</td>
<td>(&quot;key&quot;,&quot;unique&quot;,&quot;keyRef&quot;)</td>
</tr>
<tr>
<td>category</td>
<td></td>
<td></td>
</tr>
<tr>
<td>selector</td>
<td>XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>fields</td>
<td>Sequence of XPath Property Record</td>
<td></td>
</tr>
<tr>
<td>referenced key</td>
<td>(For keyRef only): Identity Constraint</td>
<td>The corresponding key constraint</td>
</tr>
</tbody>
</table>

### Model Group

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Model Group&quot;</td>
</tr>
<tr>
<td>compositor</td>
<td>string</td>
<td>(&quot;sequence&quot;,&quot;choice&quot;,&quot;all&quot;)</td>
</tr>
<tr>
<td>particles</td>
<td>Sequence of Particle</td>
<td></td>
</tr>
</tbody>
</table>

### Model Group Definition

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property type</th>
<th>Property value</th>
</tr>
</thead>
</table>
### Kind

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Model Group Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the model group</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the model group</td>
</tr>
<tr>
<td>model group</td>
<td></td>
<td>Model Group</td>
</tr>
</tbody>
</table>

### Notation

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Notation Declaration&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the notation</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the notation</td>
</tr>
<tr>
<td>system identifier</td>
<td>anyURI</td>
<td></td>
</tr>
<tr>
<td>public identifier</td>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

### Particle

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Particle&quot;</td>
</tr>
<tr>
<td>min occurs</td>
<td>integer</td>
<td></td>
</tr>
<tr>
<td>max occurs</td>
<td>integer, or string(&quot;unbounded&quot;)</td>
<td></td>
</tr>
<tr>
<td>term</td>
<td></td>
<td>Element Declaration, Element Wildcard, or ModelGroup</td>
</tr>
</tbody>
</table>

### Simple Type

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>kind</td>
<td>string</td>
<td>&quot;Simple Type Definition&quot;</td>
</tr>
<tr>
<td>name</td>
<td>string</td>
<td>Local name of the type (empty if anonymous)</td>
</tr>
<tr>
<td>target namespace</td>
<td>string</td>
<td>Namespace URI of the type (empty if anonymous)</td>
</tr>
<tr>
<td>final</td>
<td>Sequence of</td>
<td></td>
</tr>
<tr>
<td></td>
<td>string(&quot;restriction&quot;</td>
<td>extension&quot;</td>
</tr>
<tr>
<td>context</td>
<td>containing component</td>
<td></td>
</tr>
<tr>
<td>base type definition</td>
<td>Simple Type</td>
<td></td>
</tr>
<tr>
<td>facets</td>
<td>Sequence of Facet</td>
<td></td>
</tr>
<tr>
<td>fundamental facets</td>
<td>Empty sequence (not implemented)</td>
<td></td>
</tr>
<tr>
<td>variety</td>
<td>string (&quot;atomic&quot;</td>
<td>&quot;list&quot;</td>
</tr>
</tbody>
</table>
27.1.1.6 XPath/XQuery Functions: Sequence

Altova’s sequence extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of [altova:]. For example: **add-years-to-date [altova:].**
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function **without** any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: **add-years-to-date(xs:date("2014-01-15"), 10).**

**XPath functions (used in XPath expressions in XSLT):**  XP1  XP2  XP3.1

**XSLT functions (used in XPath expressions in XSLT):** XSLT1  XSLT2  XSLT3
XQuery functions (used in XQuery expressions in XQuery):

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 r 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("MyAttr", "rip") returns MyAttribute()*
- attributes("MyAttributes", "rfip") returns no match
- attributes("MyAttribute", ") returns MyAttribute()*
- attributes("MyAttributes", "Rip") returns an unrecognized-flag error.
- attributes("MyAttribute", ) returns a missing-second-argument error.

elements [altova:]

elements(ElementName as xs:string) as element() XP3.1 XQ3.1
Returns all elements that have a local name which is the same as the name supplied in the input argument, ElementName. The search is case-sensitive and conducted along the child:: axis. The context node must be the parent node of the element/s being searched for.

Examples
- elements("MyElement") returns MyElement()*

elements(ElementName as xs:string, SearchOptions as xs:string) as element() XP3.1 XQ3.1
Returns all elements that have a local name which is the same as the name supplied in the input argument, ElementName. The search is case-sensitive and conducted along the child:: axis. The context node must be the parent node of the element/s being searched for. The second argument is a string containing option flags. Available flags are:

- r = switches to a regular-expression search; ElementName must then be a regular-expression search string;
- f = If this option is specified, then ElementName provides a full match; otherwise ElementName need only partially match an element name to return that element. For example: if f is not specified, then MyElem will return MyElement;
- i = switches to a case-insensitive search;
- p = includes the namespace prefix in the search; ElementName should then contain the namespace prefix, for example: altova:MyElement.

The flags can be written in any order. Invalid flags will generate errors. One or more flags can be omitted. The empty string is allowed, and will produce the same effect as the function having only one argument (previous signature). However, an empty sequence is not allowed.

**Examples**

- elements("MyElement", "rip") returns MyElement()*
- elements("MyElement", "pri") returns MyElement()*
- elements("MyElement", ") returns MyElement()*
- elements("MyElement", "rip") returns MyElement()*
- elements("MyElements", "rfip") returns no match
- elements("MyElement", "Rip") returns an unrecognized-flag error.
- elements("MyElement", ) returns a missing-second-argument error.

**find-first** [altova:]

```
find-first((Sequence as item()*), (Condition( Sequence-Item as xs:boolean)) as item()? XP3.1 XQ3.1
```

This function takes two arguments. The first argument is a sequence of one or more items of any datatype. The second argument, Condition, is a reference to an XPath function that takes one argument (has an arity of 1) and returns a boolean. Each item of Sequence is submitted, in turn, to the function referenced in Condition. (Remember: This function takes a single argument.) The first Sequence item that causes the function in Condition to evaluate to true() is returned as the result of find-first, and the iteration stops.

**Examples**

- find-first(5 to 10, function($a) {$a mod 2 = 0}) returns xs:integer 6
  The Condition argument references the XPath 3.0 inline function, function(), which declares an inline function named $a and then defines it. Each item in the Sequence argument of find-first is passed, in turn, to $a as its input value. The input value is tested on the condition in the function definition ($a mod 2 = 0). The first input value to satisfy this condition is returned as the result of find-first (in this case 6).

- find-first((1 to 10), (function($a) {$a+3=7})) returns xs:integer 4

Further examples

If the file C:\Temp\Customers.xml exists:
• **find-first** ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) returns xs:string C:\Temp\Customers.xml

If the file C:\Temp\Customers.xml does not exist, and http://www.altova.com/index.html exists:


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-available() 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((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).

- \texttt{find-first-pos((2 to 10), (function($a) \{a+3=7\}))} returns \texttt{xs:integer 3}

**Further examples**

If the file \texttt{C:\Temp\Customers.xml} exists:

- \texttt{find-first-pos( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )} returns 1

If the file \texttt{C:\Temp\Customers.xml} does not exist, and \texttt{http://www.altova.com/index.html} exists:

- \texttt{find-first-pos( ("C:\Temp\Customers.xml", "http://www.altova.com/index.html"), (doc-available#1) )} returns 2

If the file \texttt{C:\Temp\Customers.xml} does not exist, and \texttt{http://www.altova.com/index.html} also does not exist:

- \texttt{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, \texttt{doc-available}, takes a single string argument, which is used as a URI, and returns \texttt{true} if a document node is found at the submitted URI. (The document at the submitted URI must therefore be an XML document.)
- The \texttt{doc-available} function can be used for \texttt{Condition}, the second argument of \texttt{find-first-pos}, because it takes only one argument (arity=1), because it takes an \texttt{item()} as input (a string which is used as a URI), and returns a boolean value.
- Notice that the \texttt{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 \texttt{doc-available#1} simply means: \textit{Use the doc-available() 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 \texttt{doc-available()}, which uses the string as a URI and tests whether a document node exists at the URI. If one does, the \texttt{doc-available()} function evaluates to \texttt{true()} and the index position of that string in the sequence is returned as the result of the \texttt{find-first-pos} function.
- \textit{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.

\textbf{for-each.attribute-pair [altova:]}

\begin{verbatim}
for-each.attribute-pair(Seq1 as element()?, Seq2 as element()?, Function as function()) as item() XP3.1 XQ3.1
\end{verbatim}

The first two arguments identify two elements, the attributes of which are used to build attribute pairs, where one attribute of a pair is obtained from the first element and the other attribute is obtained from the second element. Attribute pairs are selected on the basis of having the same name, and the pairs are ordered alphabetically (on their names) into a set. If, for one attribute no corresponding attribute on the other element exists, then the pair is “disjoint”, meaning that it consists of one member only. The function
item (third argument Function) is applied separately to each pair in the sequence of pairs (joint and disjoint), resulting in an output that is a sequence of items.

**Examples**

- `for-each-attribute-pair` (/Example/Test-A, /Example/Test-B, function($a, $b){$a+$b})

  returns ... 

  (2, 4, 6) if
  `<Test-A att1="1" att2="2" att3="3" />`
  `<Test-B att1="1" att2="2" att3="3" />`

  (2, 4, 6) if
  `<Test-A att2="2" att1="1" att3="3" />`
  `<Test-B att3="3" att2="2" att1="1" />`

  (2, 6) if
  `<Test-A att4="4" att1="1" att3="3" />`
  `<Test-B att3="3" att2="2" att1="1" />`

  Note: The result (2, 6) is obtained by way of the following action: (1+1, ()+2, 3+3, 4+()). If one of the operands is the empty sequence, as in the case of items 2 and 4, then the result of the addition is an empty sequence.

- `for-each-attribute-pair` (/Example/Test-A, /Example/Test-B, concat#2)

  returns ...

  (11, 22, 33) if
  `<Test-A att1="1" att2="2" att3="3" />`
  `<Test-B att1="1" att2="2" att3="3" />`

  (11, 2, 33, 4) if
  `<Test-A att4="4" att1="1" att3="3" />`
  `<Test-B att3="3" att2="2" att1="1" />`

**for-each-combination [altova:]**

`for-each-combination(FirstSequence as item()*, SecondSequence as item()*, Function($i,$j){$i || $j}) as item()*` XP3.1 XQ3.1

The items of the two sequences in the first two arguments are combined so that each item of the first sequence is combined, in order, once with each item of the second sequence. The function given as the third argument is applied to each combination in the resulting sequence, resulting in an output that is a sequence of items (see example).

**Examples**

- `for-each-combination` ( ('a', 'b', 'c'), ('1', '2', '3'), function($i, $j){$i || $j} ) returns ('a1', 'a2', 'a3', 'b1', 'b2', 'b3', 'c1', 'c2', 'c3')

**for-each-matching-attribute-pair [altova:]**

`for-each-matching-attribute-pair(Seq1 as element()?, Seq2 as element()?, Function as function()) as item()*` XP3.1 XQ3.1

The first two arguments identify two elements, the attributes of which are used to build attribute pairs,
where one attribute of a pair is obtained from the first element and the other attribute is obtained from the second element. Attribute pairs are selected on the basis of having the same name, and the pairs are ordered alphabetically (on their names) into a set. If, for one attribute no corresponding attribute on the other element exists, then no pair is built. The function item (third argument Function) is applied separately to each pair in the sequence of pairs, resulting in an output that is a sequence of items.

**Examples**

- **for-each-matching-attribute-pair** (/Example/Test-A, /Example/Test-B, function($a, $b){$a+$b}) returns ...

  \[
  \begin{align*}
  (2, 4, 6) \text{ if } & \begin{cases} 
  \text{<Test-A att1="1" att2="2" att3="3" />} \\
  \text{<Test-B att1="1" att2="2" att3="3" />}
  \end{cases} \\
  (2, 4, 6) \text{ if } & \begin{cases} 
  \text{<Test-A att2="2" att1="1" att3="3" />} \\
  \text{<Test-B att3="3" att2="2" att1="1" />}
  \end{cases} \\
  (2, 6) \text{ if } & \begin{cases} 
  \text{<Test-A att4="4" att1="1" att3="3" }} \\
  \text{<Test-B att3="3" att2="2" att1="1" />}
  \end{cases}
  \end{align*}
  \]

- **for-each-matching-attribute-pair** (/Example/Test-A, /Example/Test-B, concat#2) returns ...

  \[
  \begin{align*}
  (11, 22, 33) \text{ if } & \begin{cases} 
  \text{<Test-A att1="1" att2="2" att3="3" }} \\
  \text{<Test-B att1="1" att2="2" att3="3" />}
  \end{cases} \\
  (11, 33) \text{ if } & \begin{cases} 
  \text{<Test-A att4="4" att1="1" att3="3" }} \\
  \text{<Test-B att3="3" att2="2" att1="1" />}
  \end{cases}
  \end{align*}
  \]

\[
\text{ substitute-empty [altova:]} \quad \text{substitute-empty(FirstSequence as item(*)*, SecondSequence as item()) as item()* XP3.1 XQ3.1}
\]

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)

\[
\text{27.1.1.7 XPath/XQuery Functions: String}
\]

Altova's string extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.
Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova
  extension functions are named in this documentation with a suffix of [altova:]. For example: `add-
  years-to-date [altova:]`. When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function without any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.

<table>
<thead>
<tr>
<th>XPath functions (used in XPath expressions in XSLT):</th>
<th>XP1</th>
<th>XP2</th>
<th>XP3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSLT functions (used in XPath expressions in XSLT):</td>
<td>XSLT1</td>
<td>XSLT2</td>
<td>XSLT3</td>
</tr>
<tr>
<td>XQuery functions (used in XPath expressions in XQuery):</td>
<td>XQ1</td>
<td>XQ3.1</td>
<td></td>
</tr>
</tbody>
</table>

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.

\[
\text{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.

\begin{itemize}
  \item `char("2014-01-15", 5)` returns -
  \item `char("USA", 1)` returns U
  \item `char("USA", 10)` returns the empty string.
  \item `char("USA", -2)` returns the empty string.
\end{itemize}

\begin{itemize}
  \item `create-hash-from-string('abc')` returns a hash string generated by using the SHA-256 hashing algorithm.
  \item `create-hash-from-string('abc', 'md5')` returns a hash string generated by using the MD5 hashing algorithm.
  \item `create-hash-from-string('abc', 'MD5')` returns a hash string generated by using the MD5 hashing algorithm.
\end{itemize}

\begin{itemize}
  \item `first-chars(X-Number as \(xs:integer\)) as \(xs:string\) XP3.1 XQ3.1` 
  \item `first-chars(2)` returns 12
  \item `first-chars(5)` returns 1234A
  \item `first-chars(9)` returns 1234ABCD
\end{itemize}
first-chars(InputString as xs:string, X-Number as xs:integer) as xs:string XP3.1 XQ3.1

Returns a string containing the first X-Number of characters of the string submitted as the InputString argument.

Examples

- first-chars("2014-01-15", 5) returns 2014-
- first-chars("USA", 1) returns U

format-string [altova:]

format-string(InputString as xs:string, FormatSequence as item()* ) as xs:string XP3.1 XQ3.1

The input string (first argument) contains positional parameters (%1, %2, etc). Each parameter is replaced by the string item that is located at the corresponding position in the format sequence (submitted as the second argument). So the first item in the format sequence replaces the positional parameter %1, the second item replaces %2, and so on. The function returns this formatted string that contains the replacements. If no string exists for a positional parameter, then the positional parameter itself is returned. This happens when the index of a positional parameter is greater than the number of items in the format sequence.

Examples

- format-string ('Hello %1, %2, %3', (Jane, 'John', 'Joe')) returns "Hello Jane, John, Joe"
- format-string ('Hello %1, %2, %3', (Jane, 'John', 'Joe', 'Tom')) returns "Hello Jane, John, Joe, Tom"
- format-string ('Hello %1, %2, %4', (Jane, 'John', 'Joe', 'Tom')) returns "Hello Jane, John, Tom"
- format-string ('Hello %1, %2, %4', (Jane, 'John', 'Joe')) returns "Hello Jane, John, %4"

last-chars [altova:]

last-chars(X-Number as xs:integer) as xs:string XP3.1 XQ3.1

Returns a string containing the last X-Number of characters of the string obtained by converting the value of the context item to xs:string.

Examples

If the context item is 1234ABCD:

- last-chars(2) returns CD
- last-chars(5) returns 4ABCD
- last-chars(9) returns 1234ABCD

last-chars(InputString as xs:string, X-Number as xs:integer) as xs:string XP3.1 XQ3.1

Returns a string containing the last X-Number of characters of the string submitted as the InputString argument.

Examples

- last-chars("2014-01-15", 5) returns 01-15
- last-chars("USA", 10) returns USA
pad-string-left [altova:]

```
pad-string-left(StringToPad as xs:string, StringLength as xs:integer, PadCharacter as xs:string) as xs:string XP3.1 XQ3.1
```

The PadCharacter argument is a single character. It is padded to the left of the string to increase the number of characters in StringToPad so that this number equals the integer value of the StringLength argument. The StringLength argument can have any integer value (positive or negative), but padding will occur only if the value of StringLength is greater than the number of characters in StringToPad. If StringToPad has more characters than the value of StringLength, then StringToPad is left unchanged.

**Examples**

- `pad-string-left('AP', 1, 'Z')` returns `'AP'
- `pad-string-left('AP', 2, 'Z')` returns `'AP'
- `pad-string-left('AP', 3, 'Z')` returns `'ZAP'
- `pad-string-left('AP', 4, 'Z')` returns `'ZAP'
- `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
```

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**

- \texttt{substring-after-last('ABCDEFGH', 'B')} returns \texttt{CDEFGH}
- \texttt{substring-after-last('ABCDEFGH', 'BC')} returns \texttt{DEFGH}
- \texttt{substring-after-last('ABCDEFGH', 'BD')} returns \texttt{''}
- \texttt{substring-after-last('ABCDEFGH', 'Z')} returns \texttt{''}
- \texttt{substring-after-last('ABCDEFGH', '')} returns \texttt{ABCDEFGH}
- \texttt{substring-after-last('ABCD-ABCD', 'B')} returns \texttt{CD}
- \texttt{substring-after-last('ABCD-ABCD-ABCD', 'BCD')} returns \texttt{''}

\[ \text{substring-before-last} \]

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**

- \texttt{substring-before-last('ABCDEFGH', 'B')} returns \texttt{A}
- \texttt{substring-before-last('ABCDEFGH', 'BC')} returns \texttt{A}
- \texttt{substring-before-last('ABCDEFGH', 'BD')} returns \texttt{''}
- \texttt{substring-before-last('ABCDEFGH', 'Z')} returns \texttt{''}
- \texttt{substring-before-last('ABCDEFGH', '')} returns \texttt{''}
- \texttt{substring-before-last('ABCD-ABCD', 'B')} returns \texttt{ABCD-A}
- \texttt{substring-before-last('ABCD-ABCD-ABCD', 'ABCD')} returns \texttt{ABCD-ABCD-''}

\[ \text{substring-pos} \]

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**

- \texttt{substring-pos('Altova', 'to')} returns \texttt{3}
- \texttt{substring-pos('Altova', 'tov')} returns \texttt{3}
- \texttt{substring-pos('Altova', 'tv')} returns \texttt{0}
- \texttt{substring-pos('AltovaAltova', 'to')} returns \texttt{3}
- \texttt{substring-pos('AltovaAltova', 'to')} returns \texttt{3}
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-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-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"
27.1.1.8 XPath/XQuery Functions: Miscellaneous

These miscellaneous extension functions can be used in XPath and XQuery expressions and provide additional functionality for the processing of data.

Note about naming of functions and language applicability

Altova extension functions can be used in XPath/XQuery expressions. They provide additional functionality to the functionality that is available in the standard library of XPath, XQuery, and XSLT functions.

- In order to distinguish Altova extension functions from functions in the standard library, Altova extension functions are named in this documentation with a suffix of [altova:]. For example: `add-years-to-date [altova:]`.
- When using Altova extension functions in your XPath/XQuery expressions, however, you must use the function without any prefix or suffix, just as you would use any standard XPath/XQuery function. Use an Altova extension like this: `add-years-to-date(xs:date("2014-01-15"), 10)`.


| XPath functions (used in XPath expressions in XSLT): | XP1 XP2 XP3.1 |
| XSLT functions (used in XPath expressions in XSLT): | XSLT1 XSLT2 XSLT3 |
| XQuery functions (used in XQuery expressions in XQuery): | XQ1 XQ3.1 |

- **decode-string [altova:]**
  
  `decode-string(Input as xs:base64Binary) as xs:string`  
  `decode-string(Input as xs:base64Binary, Encoding as xs:string) as xs:string`  
  Decodes the submitted base64Binary input to a string using the specified encoding. If no encoding is specified, then the UTF-8 encoding is used. The following encodings are supported: US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4

- **Examples**
  
  - `decode-string($XML1/MailData/Meta/b64B)` returns the base64Binary input as a UTF-8 encoded string
  - `decode-string($XML1/MailData/Meta/b64B, "UTF-8")` returns the base64Binary input as a UTF-8-encoded string
  - `decode-string($XML1/MailData/Meta/b64B, "ISO-8859-1")` returns the base64Binary input as an ISO-8859-1-encoded string

- **encode-string [altova:]**
  
  `encode-string(InputString as xs:string) as xs:base64Binaryinteger`  
  Encodes the string input into a base64Binary format.

- **trim-string-right**
  
  `trim-string-right("Hello World")` returns "Hello World"
  `trim-string-right("Hello   World")` returns "Hello   World"
encode-string(InputString as xs:string, Encoding as xs:string) as xs:base64BinaryInteger XP2 XP3.1 XQ3.1

Encodes the submitted string using, if one is given, the specified encoding. If no encoding is given, then the UTF-8 encoding is used. The encoded string is converted to base64Binary characters, and the converted base64Binary value is returned. Initially, UTF-8 encoding is supported, and support will be extended to the following encodings: US-ASCII, ISO-8859-1, UTF-16, UTF-16LE, UTF-16BE, ISO-10646-UCS2, UTF-32, UTF-32LE, UTF-32BE, ISO-10646-UCS4

Examples

- `encode-string("Altova")` returns the base64Binary equivalent of the UTF-8 encoded string "Altova"
- `encode-string("Altova", "UTF-8")` returns the base64Binary equivalent of the UTF-8 encoded string "Altova"

generate-guid [altova:]

generate-guid() as xs:string XP2 XP3.1 XQ2 XQ3.1

Generates a unique string GUID string.

Examples

- `generate-guid()` returns (for example) 85F971DA-17F3-4E4E-994E-99137873ACCD

high-res-timer [altova:]

high-res-timer() as xs:double XP3.1 XQ3.1

Returns a system high-resolution timer value in seconds. A high-resolution timer, when present on a system, enables high precision time measurements when these are required (for example, in animations and for determining precise code-execution time). This function provides the resolution of the system's high-res timer.

Examples

- `high-res-timer()` returns something like '1.16766146154566E6'

parse-html [altova:]

parse-html(HTMLText as xs:string) as node() XP3.1 XQ3.1

The HTMLText argument is a string that contains the text of an HTML document. The function creates an HTML tree from the string. The submitted string may or may not contain the HTML element. In either case, the root element of the tree is an element named HTML. It is best to make sure that the HTML code in the submitted string is valid HTML.

Examples

- `parse-html("<html><head/><body><h1>Header</h1></body></html>")` creates an HTML tree from the submitted string

sleep[altova:]

sleep(Millisecs as xs:integer) as empty-sequence() XP2 XP3.1 XQ3.1

Suspends execution of the current operation for the number of milliseconds given by the Millisecs
argument.

**Examples**

- \texttt{sleep(1000)} suspends execution of the current operation for 1000 milliseconds.
27.2 License Information

This section contains information about:

- the distribution of this software product
- software activation and license metering
- the license agreement governing the use of this product

Please read this information carefully. It is binding upon you since you agreed to these terms when you installed this software product.

To view the terms of any Altova license, go to the Altova Legal Information page at the Altova website.

27.2.1 Electronic Software Distribution

This product is available through electronic software distribution, a distribution method that provides the following unique benefits:

- You can evaluate the software free-of-charge for 30 days before making a purchasing decision. *(Note: Altova MobileTogether Designer is licensed free of charge.)*
- Once you decide to buy the software, you can place your order online at the Altova website and get a fully licensed product within minutes.
- When you place an online order, you always get the latest version of our software.
- The product package includes an onscreen help system that can be accessed from within the application interface. The latest version of the user manual is available at www.altova.com in (i) HTML format for online browsing, and (ii) PDF format for download (and to print if you prefer to have the documentation on paper).

Helping Others within Your Organization to Evaluate the Software

If you wish to distribute the evaluation version within your company network, or if you plan to use it on a PC that is not connected to the Internet, you may distribute only the installer file, provided that this file is not modified in any way. Any person who accesses the software installer that you have provided must request their own 30-day evaluation license key code and after expiration of their evaluation period, must also purchase a license in order to be able to continue using the product.

27.2.2 Software Activation and License Metering

As part of Altova’s Software Activation, the software may use your internal network and Internet connection for the purpose of transmitting license-related data at the time of installation, registration, use, or update to an Altova-operated license server and validating the authenticity of the license-related data in order to protect Altova against unlicensed or illegal use of the software and to improve customer service. Activation is based on the exchange of license related data such as operating system, IP address, date/time, software version, and computer name, along with other information between your computer and an Altova license server.
Your Altova product has a built-in license metering module that further helps you avoid any unintentional violation of the End User License Agreement. Your product is licensed either as a single-user or multi-user installation, and the license-metering module makes sure that no more than the licensed number of users use the application concurrently.

This license-metering technology uses your local area network (LAN) to communicate between instances of the application running on different computers.

**Single license**

When the application starts up, as part of the license metering process, the software sends a short broadcast datagram to find any other instance of the product running on another computer in the same network segment. If it doesn't get any response, it will open a port for listening to other instances of the application.

**Multi-user license**

If more than one instance of the application is used within the same LAN, these instances will briefly communicate with each other on startup. These instances exchange key-codes in order to help you to better determine that the number of concurrent licenses purchased is not accidentally violated. This is the same kind of license metering technology that is common in the Unix world and with a number of database development tools. It allows Altova customers to purchase reasonably-priced concurrent-use multi-user licenses.

We have also designed the applications so that they send few and small network packets so as to not put a burden on your network. The TCP/IP ports (2799) used by your Altova product are officially registered with the IANA (see the [IANA Service Name Registry](https://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.

**Note about certificates**

Your Altova application contacts the Altova licensing server ([link.altova.com](http://link.altova.com)) via HTTPS. For this communication, Altova uses a registered SSL certificate. If this certificate is replaced (for example, by your IT department or an external agency), then your Altova application will warn you about the connection being insecure. You could use the replacement certificate to start your Altova application, but you would be doing this at your own risk. If you see a *Non-secure connection* warning message, check the origin of the certificate and consult your IT team (who would be able to decide whether the interception and replacement of the Altova certificate should continue or not).

If your organization needs to use its own certificate (for example, to monitor communication to and from client machines), then we recommend that you install Altova's free license management software, Altova [LicenseServer](https://www.altova.com/products/license-server), on your network. Under this setup, client machines can continue to use your organization's certificates, while Altova LicenseServer can be allowed to use the Altova certificate for communication with Altova.
27.2.3 Altova MobileTogether Designer End User License Agreement

- The Altova MobileTogether Designer End-User License Agreement is available here: https://www.altova.com/legal/mobiletogether-eula
- Altova's Privacy Policy is available here: https://www.altova.com/privacy
Index

$MT_Global variables (dynamic), 1046
$MT_Global variables (static), 1042
$MT_AuthenticationToken, 978
$MT_DBSTRUCTURE, 740
$MT_EMBEDDEDMESSAGE page source, for UWP apps, 1369
$MT_IN_APP_PURCHASE page source, 1337
$MT_SERVICE page source, 1373
in simulations, 1216
$PERSISTENT tree, 1223, 1225, 1228

.mtp files, 239, 1389

A

Acknowledge Purchase action, 671, 1341, 1343
Acknowledge Purchase action (in example project), 1358
Action Groups, 795
calling, to set variable value of Let action, 806
creating and editing, 798
using, 799
Actions, 574
Access Calendars, 607
Acknowledge Purchase, 671
Append Node, 770
Audio (playback), 624
Audio Recording, 628
Backup/Restore Page Sources, 727
Backup/Restore SQLite DB, 743
Break Loop, 785
Cancel Action Execution, 750
Close Subpage, 682
Comment, 748
DB Begin Transaction, 730
DB Bulk Insert Into, 735
DB Commit Transaction, 737
DB Execute, 732
DB Read Structure, 740
DB Rollback Transaction, 739
Delete Node, 775
Execute at Once, 750
Execute FlowForce Job, 722
Execute On, 748
Execute REST Request, 721
Execute SOAP Request, 719
ExecuteOn, 850
for OnPageLoad, 79
Get File Info, 723
Get/Report Consumable, 672
Go to Page, 677
Go to Subpage, 677
Hide Keyboard, 684
If-Then, 782
If-Then-Else, 782
Insert Node, 767
Let, 786
Let User Choose Date, 611
Let User Choose Image, 615
Let User Choose Time, 612
Let User Scan Barcode, 623
Load from binary file, 696
Load from file, 690
Load from String, 710
Load from text file to node, 702
Load/Save File, 690
Load/Save from SOAP, 713
Load/Save HTTP/FTP, 708
Loop, 783
Make Call To, 591
MapForce Transfer, 602
Measure Controls, 756
Message Box, 580
Open file, 591
Open URL, 89, 591
Print To, 596
Purchase, 668
Query Available Products, 670
Query Purchases (Android, Windows), 669
Read Contacts, 605
Read Folder, 725
Read Geo Data, 640
Reload, 689
Index 1573

Actions, 574
Replace Node, 776
Reset, 718
Restore Purchases (iOS only), 669
Return, 793
Save, 91, 715
Save from node to text file, 702
Save Image to File, 616
Save to binary file, 696
Save to file, 690
Save to String, 710
Scroll to Bottom, 682
Send Email To, 582
Send SMS To, 590
setting for events, 70
Share, 587
Show Geolocation on Map, 644
Show/Hide Wait Cursor, 605
Solution Execution, 753
Start Geolocation Tracking, 639
Stop Geolocation Tracking, 639
Text to Speech, 631
Throw, 789
Try/Catch Exceptions, 791
Try/Catch Server Connection Error, 793
Update Display, 684
Update Node, 762
Update Variable, 789
User Cancel Behavior, 751
Video (playback), 632
Video Recording, 634
View Image, 622
Actions Debugger, 1232
selecting actions to debug, 1234
addEventListener(), 1252
Address book sample file, 1214
ADO,
as data connection interface, 1058
setting up a connection, 1064
ADO.NET,
setting up a connection, 1070
Alias,
see Global Resources, 1176
Altova extensions,
chart functions (see chart functions), 1493
Altova Global Resources,
see under Global Resources, 1176
Android Beam, 651
Anonymous login, 1254
Append Node action, 770
area chart features, 914
Assertion Message control, 345
Assertions, 345
Audio,
file formats, encoding, 860
playback action, 624
recording action, 628
Audio playback,
overview of, 852
starting, pausing, resuming, stopping, skipping to, 624
Audio recording,
overview of, 854
starting, stopping, 628
Authenticated users, 978
Authentication for Embedded Webpage Solutions, 1254
anonymous login, 1254
JWT authentication, 1255
user login, 1255
Authentication token, 978
Auto-fit groups listing, 1444
Automated testing, 1287
comparing test runs, 1302
deploying test cases to server, 1298
managing test cases and runs, 1294
playing back test cases, 1291
recording test cases, 1289

B
Backup/Restore SQLite DB action, 743
bar chart features, 914
Barcodes, 623
Base64-encoded images,
see under Images, 835
Border settings for tables, 1452
Break conditions in XPath Debugger, 996
Break Loop action, 785
Breakpoints in XPath Debugger, 996
Breakpoints Pane, 214
Button control,
OnButtonClicked, 352
Caches, 316
  creating and configuring, 317
  editing configuration of, 318
  overview and mapping of all server caches, 318
Caching, 254
Calendar sample files, 1215
Calendars,
  read data into page source, 607
  read event information from, 607
  write event information to, 607
Callstacks in action debugging, 1234
Cancel Action Execution action, 750
candlestick chart features, 914
Chart control, 371
Chart data storage and transmission, 254, 287
Charts, 879
  3d settings, 925
  adding legend, 912
  appearance, 901
  area chart features, 914
  background color, 912
  bar chart features, 914
  candlestick chart features, 914
  color range, 918
  color schema, 918
  creating and configuring, 879
  data selection, 882
  data selection: flexible, 891
  data selection: simple, 886
  defining colors, 918
  fonts, 927
  gauge chart features, 914
  grid lines, 920, 923, 924
  line chart features, 914
  margins, 925
  pie chart features, 914
  removing legend, 912
  series color, 918
  sizes, 925
  tick size, 925
  title, 912
  X-axis, 920
  Y-axis, 923
Z-axis, 924
Charts tutorial, 119, 130
Check Box control, 379
Client trial run settings, 1474
Client-side data storage, 256
Close Subpage action, 682
Combo Box,
  adding and defining, 65
  drop down list definitions, 127
  editing drop down list, 85
  events and actions, 70
  source node of, 82
Combo Box control, 390
Comment action, 748
Configurations,
  of a global resource, 1177
Configurations in global resources, 1188
Contacts from address book,
  reading, 605
  storing as a data source, 605
Contacts sample files, 1214
Control actions, 70
Control events, 70, 572
  and their actions, 572
Control templates, 930
  adding via Placeholder Controls, 480
  array expressions in, 950
  changing context node of, 480
  content of, 931
  context node for evaluations in, 931
  creating, 931
  defining parameter values for, 480
  example of creating and using, 942
  example of dynamic modification, 948
  example of dynamic selection, 950
  example of parameter use, 948, 950
  example projects, 939
  map expressions in, 950
  modifying on placeholder, 937
  parameters and variables, 931
  passing values to parameters of, 950
  switching between, in placeholder calls, 946
  use first visible control, 931
  use of variables in, 946
  using in page layouts, 935
Control widths,
  measurement of, 756
Controls, 340

© 2015-2021 Altova GmbH
Controls, 340
  Assertion message, 345
  Button, 352
  Chart, 371
  Check Box, 379
  Combo Box, 390
  common context menu commands of, 341
  Date, 403
  DateTime (iOS), 414
  Edit Field, 424
  Geolocation Map, 436
  Horizontal Line, 443, 559
  Horizontal Slider, 448
  Image, 455
  Label, 467
  Placeholder Control, 480
  Radio Button, 482
  Rich Text, 494
  Signature Field, 500
  Space, 510
  Switch, 512
  Table, 523
  Time, 549
  Video, 563

Controls Pane, 211
Copyright information, 1569
Custom HTTP verbs in REST requests, 270

Database drivers,
  overview, 1061

Database tutorial, 119

Databases, 735, 1052
  actions for, 728
  and auto-increment of fields, 1054
  and DB Execute action, 1154
  and display of data in MobileTogether, 1158
  and global resources, 1187
  and tables, 129, 135, 141
  as data sources, 1054
  backing up and restoring SQLite DBs, 743
  committing transactions, 737
  editing DB data, 1146
  enabling editing of, 135, 141
  executing SQL statements, 732
  images in, 851
  NULL values, 715
  OriginalRowSet, 1054
  primary key values, 1054
  reading to SMT_DBSTRUCTURE, 740
  rolling back transactions, 739
  saving data to, 1150
  selecting data sources with SELECT statements, 1143
  selecting DB objects as data sources, 1143
  selecting tables as datasources, 1143
  starting transactions, 730

DatabaseSpy,
  3d charts, 925
  area chart features, 914
  bar chart features, 914
  candlestick chart features, 914
  chart background, 912
  chart colors, 918
  chart font options, 927
  chart fonts, 927
  chart grid, 920, 923, 924
  chart legend, 912
  chart title, 912
  chart X-axis, 920
  chart Y-axis, 923
  chart Z-axis, 924
  charts sizes, 925
  gauge chart features, 914
  line chart features, 914
  pie chart features, 914

Data control, 403

Dates,
Dates,
user selection of, 611

DateTime control (iOS), 414

DB,
see Databases, 1052

DB Begin Transaction action, 730

DB Commit Transaction action, 737

DB data sources, 123

DB Execute action, 732
  example of use, 1223

DB page source, 281

DB Query View, 205

DB Rollback Transaction action, 739

DB transactions, 730, 737, 739

Debugger for actions and XPath expressions, 1232

Debugger Modes in Simulator, 1233

Debugging design actions, 1234

Debugging XPath expressions, 996

Default file, 87

Default file of data source trees, 296

Default XPath context node, 61

Defining,
  3d settings, 925
  area chart features, 914
  bar chart features, 914
  candlestick chart features, 914
  chart fonts, 927
  charts colors, 918
  charts sizes, 925
  charts title, 912
  color of charts, 918
  fonts in charts, 927
  gauge chart features, 914
  grid lines, 920, 923, 924
  line chart features, 914
  pie chart features, 914
  X-axis settings, 920
  Y-axis settings, 923
  Z-axis settings, 924

Delete Node action, 775

Delete Node(s),
  example of use, 1225

Delete Node(s) action,
  example of use, 1228

Deploy to server, 76

Deploying project to server,
  files to deploy, 209

Deploying solutions,
  MobileTogether Server settings for, 1474

Deploying test cases to server, 1298

Deploying the project, 235

Design file,
  creating new, 58

Design steps, 51

Device resolution and sizes, 984

Device-independent pixels (dp) as a length unit, 984

Distribution,
of Altova's software products, 1569

dp (device-independent pixels) as a length unit, 984

E

Edit Field control, 424

Edit XPath/XQuery Expression Dialog, 988

Email, 582

Email settings, 1481

Embed XML in design, 254

Embedded Message Back action, 758

Embedded messages in UWP apps, 1369

Embedded solutions, 1243
  accessing and sending XML data, 710
  addEventListener(), 1252
  automatic load of JSON page source, 1277
  communication with webpage, 1250
  data communication in, 1250
  example of how to embed, 1261
  examples, 1261
  how to embed in webpage, 1247
  IFrames for, 1247
  JSON example, 1261
  JWT authentication example, 1283
  messages from server to webpage, 1252
  page source, 337, 710
  posting from webpage, 1251
  postMessage(), 1251
  pre-setting webpage data to IFrame, 1277
  sendig messages, 337
  sending data from server to webpage, 1261, 1269
  sending data from webpage to server, 1261, 1269
  sending/receiving data from webpage (JSON example), 1261
  sending/receiving data from webpage (XML example), 1269
  XML example, 1269

Embedded webpage solutions,
  see Embedded solutions, 1243
Index

EmbeddedMessage page source, 337
EmbeddedMessageBack action, for UWP apps, 1369
Embedding a solution in a webpage, 1261
End User License Agreement, 1569
Enter shortcut, displaying for controls on page, 1442
Enter shortcut, 352, 371, 455, 467
Error messages, 758
Escape shortcut, 352, 371, 455, 467 displaying for controls on page, 1442
Evaluation period, of Altova's software products, 1569
Events, control events, 572 page events, 329 setting actions for, 70
Events in calendars, reading and writing, 607
Execute at Once action, 750
Execute On action, 748
ExecuteOn, 850
Exif images, see under Images, 838
Expression Builder, 989
External PN keys, 662

FTP, adding as page source, 268 settings for requests, 268
FTP page source, 267

Ga
gauge chart features, 914
Geolocation actions, 637
Geolocation data, entering in $GEOLOCATION tree, 640
Geolocation Map control, 436
Geolocation tracking, starting and stopping, 639
Get Products Available action, 1339
Get Products Available action (in example project), 1350, 1351
Get/Report Consumable action, 672
Get/Report Consumables action, 1343
Global Resources, 253, 1176 changing configurations, 1188 defining, 1177 defining database-type, 1183 defining file-type, 1179 defining folder-type, 1182 using, 1186, 1187, 1188
Global Resources XML File, 1177
Global variables, 1040
  MT_Android, 1042
  MT_AudioChannel, 1046
  MT_AuthenticationToken, 1042
  MT_Browser, 1042
  MT_ButtonBackgroundColor, 1042
  MT_ButtonTextColor, 1042
  MT_CameraAvailable, 1042
  MT_CanvasX, 1046
  MT_CanvasY, 1046
  MT_ClientLanguage, 1042
  MT_ControlKind, 1046
  MT_ControlName, 1046
  MT_ControlNode, 1046
  MT_ControlValue, 1046
  MT_ControlValueBeforeChange, 1046
  MT_DBExecute_Result, 1046
  MT_DeviceHeight, 1042
  MT_DeviceWidth, 1042

F

File DSN, setting up, 1077
File locations for designer simulation, 1191
File locations for server simulation, 1196
Files Pane, 209
Find & Replace Pane, 227
Firebird, Connecting through JDBC, 1089 Connecting through ODBC, 1090
First visible control, use of, in control templates, 931
FlowForce, using to add page source, 284
FlowForce jobs, 722
Foreign keys, disable in SQLite, 1087
Formatting the design, 64
Global variables, 1040
- MT_DPIX, 1042
- MT_DPIY, 1042
- MT_EditFieldBackgroundColor, 1042
- MT_EditFieldTextColor, 1042
- MT_FirstPageLoad, 1046
- MT_GeolocationAvailable, 1042
- MT_HTTPExecute_Result, 1046
- MT_InputParameters, 1042
- MT_iOS, 1042
- MT_iPad, 1042
- MT_IsAppStoreApp, 1042
- MT_IsEmbedded, 1042
- MT_LabelBackgroundColor, 1042
- MT_LabelTextColor, 1042
- MT_Landscape, 1046
- MT_MeasureControls, 1046
- MT_NFCAvailable, 1042
- MT_PageBackgroundColor, 1042
- MT_PageName, 1046
- MT_Porat, 1046
- MT_ServerConnectionErrorLocation, 1046
- MT_SimulationMode, 1042
- MT_SMSAvailable, 1042
- MT_TableColumnContext, 1046
- MT_TargetNode, 1046
- MT_TelephonyAvailable, 1042
- MT_UserName, 1042
- MT_UserRoles, 1046
- MT_WindowHeight, 1046
- MT_Windows, 1042
- MT_WindowsPhone, 1042
- MT_WindowWidth, 1046
- user-defined, 1050

Global variables (dynamic), 1046
Global variables (static), 1042
Go to Page action, 677
Go to Subpage action, 677
GUI,
- see User interface, 201

HTML page source, 263
HTTP,
- adding as page source, 268
- settings for requests, 268
HTTP page source, 267
Hyperlinking to solutions, 980

IBM DB2,
- connecting through JDBC, 1093
- connecting through ODBC, 1095
IBM DB2 for i,
- connecting through JDBC, 1101
- connecting through ODBC, 1102

IBM Informix,
- connecting through JDBC, 1105

If-Then action, 782
If-Then-Else action, 782

Image actions, 613
Image control, 455, 834
Images, 67, 834
- Base64, 838, 844
- Base64 or URL, 834
- Base64-encoded, 835
- changing URL of, 82
- Exif, 835, 838
- file extension, 616
- in databases, 851
- resizing, 850
- rotating, 850
- saving to file, 616
- sources for, 834
- transforming, 850
- transforming on server, 748
- user-selected, 615, 844

Importing page source structure from file, 259

In-app purchases,
- $MT_IN_APP_PURCHASE page source, 1341
- $MT_UpdatedInAppPurchases variable, 1343
- $PERSISTENT page source (in example project), 1350
- Acknowledge Purchase action, 1341, 1343
- Acknowledge Purchase action (in example project), 1358
- acknowledging an in-app purchase, 1358
- categories of, 1333
- check availability of service, 1343

Hide Keyboard action, 684
Horizontal Line control, 443, 559
Horizontal Slider control, 448
In-app purchases,
data file for simulations, 1347
defining available products, 1335
defining the Purchase request, 1341, 1343, 1354
device for simulations, 1347
eexample project, 1347
functions to check app store responses, 1343
Get Products Available action (in example project), 1350, 1351
getting list of products for purchase, 1339
mapping SKU IDs to product name, 1335
mapping SKU IDs to product name (in example project), 1349
mt-last-in-app-purchase-response-code(), 1341, 1343
OnPurchaseUpdated event, 1341, 1343
OnPurchaseUpdated event (in example project), 1358
page source of, 1337
product names in project, 1335
PurchaseState property, 1343
querying/restoring purchases (in example project), 1359
service for in-app purchases, 1350
setting up for MT, 1333
simulations, 1346
testing, 1346
trial run on client, 1346
Insert Node action, 767

JWT authentication example, 1283

L

Label control, 467
Languages,
switching at runtime, 754
Legal information, 1569
Length units, 984
Let action, 786
Let User Choose Date action, 611
Let User Choose Image action, 615
Let User Choose Time action, 612
Let User Scan Barcode action, 623
License,
information about, 1569
License metering,
in Altova products, 1569
line chart features, 914
Link to MobileTogether demo videos, 1472
Linking to solutions, 980
Listings Pane, 225
Load Data setting,
example of use, 1223, 1225, 1228
significance of, 1221
Load from String, 710
Load from String action, 337
Local dynamic variables,
see Global Variables (dynamic), 1046
Localization, 250, 1423
changing language of solution, 754
in simulations, 1429
Log Message action, 758
Logo of MobileTogether,
at program start, 1472
in print output, 1472
Loop action, 783
Loops, 783, 785

M

Main Window,
description and features, 203
Make Call To action, 591
MapForce Transfer action, 602
MariaDB,
connect through ODBC, 1107
Message Box action, 580
Messages Pane, 223
MFX files, 602
Microsoft Access,
connecting through ADO, 1064, 1109
Microsoft SQL Server,
connecting through ADO, 1111
connecting through ODBC, 1114
Mixed-content elements,
serializing of, 1038
MobileTogether,
system requirements, 48
terminology, 50
MobileTogether Client, 48
MobileTogether libraries for UWP integration, 1365
MobileTogether overview, 48
MobileTogether Packages, 239, 1389
MobileTogether Server, 48
MobileTogether Server Advanced Edition, 1372
MobileTogether Server settings, 1474
Modal dialogs, 677
MT Debugger, 1232
Actions Debugger (see there also), 1234
MT Recording file, 1294
MT_Global variables (dynamic), 1046
MT_Global variables (static), 1042
MT_SERVICE page source,
see $MT_SERVICE page source, 1216
mt-in-app-purchase-platform-to-product(), 1335
mt-in-app-purchase-product-to-platform(), 1335
mt-in-app-purchase-service-started(), 1350
mt-last-file-path, 634, 859
MTP files, 239, 1389
MTPNSIM files, 877, 1212
MySQL,
connecting through ODBC, 1120

New features, 23
NFC,
and Android Beam, 651, 862
availability on devices, 862
devices, 862
events related to, 866
MobileTogether support for, 862
overview of design components for, 867
Push action, 651, 865
reading data from NFC tags, 864
sending data, 651, 865
Start/Stop action, 650
tags, 862, 864
technology, 862
NFC actions, 648
NFC sample files, 1210
NFC tree structure, 650, 864
Node data,
loading from a text file, 702
saving to a text file, 702
Nodes,
creating new, 767, 770

OAuth in REST requests, 270
ODBC,
as data connection interface, 1058
connect to MariaDB, 1107
connect to Teradata, 1138
setting up a connection, 1077
ODBC Drivers,
checking availability of, 1077
Offline use, 1221, 1223, 1225, 1228
OLE DB,
as data connection interface, 1058
OnClicked, 352, 371, 455, 467
OnEmbeddedMessage event, 337
for UWP apps, 1369
OnPageLoad,
example of use, 1228
OnPageRefresh,
example of use, 1225, 1228
OnServiceRunning, 1373
Open file action, 591
Open URL action, 89, 591

Namespaces, 252
Namespaces in the project, 293
NDEF technology, 862
Network proxy settings, 1482
OpenJDK,  as Java Virtual Machine, 1080

Options,
  3d charts, 925
  area chart features, 914
  bar chart features, 914
  candlestick chart features, 914
  chart colors, 918
  chart fonts, 927
  chart grid, 920, 923, 924
  chart legend, 912
  chart title, 912
  chart X-axis, 920
  chart Y-axis, 923
  chart Z-axis, 924
  charts background, 912
  charts sizes, 925
  gauge chart features, 914
  line chart features, 914
  pie chart features, 914

Oracle database,
  connecting through JDBC, 1122
  connecting through ODBC, 1124

OriginalRowSet, 105, 123

Outlook calendars, 1215
Outlook contacts, 1214
Overview Pane, 217

Page Controls, 211, 341
Page data source,
  adding, 61
Page design, 320
Page Design View, 204
Page events, 329
  and their actions, 329
Page properties, 324
Page sequence, 120
Page setup, 60
Page source link, 82
Page source trees, 289
  context menus of, 301
  importing data from file, 296
  importing XML structure into, 294
  read-only and editable, 296
Page sources,
  adding a DB source, 281
  adding a JSON source, 265
  adding an HTML source, 263
  adding an XML source, 261
  adding an XQuery source, 282
  adding via a FlowForce job, 284
  backing up temporarily to internal memory, 727
  importing structure from file, 259
  options, 287
  restoring/discarding internally backed up, 727
  reusing across pages, 259
  root nodes of, 291
  trees, their structure and data, 289

Page Sources actions, 687
Page Sources Pane,
  creating a tree structure, 215
  features of, 215
Page View settings, 204
Page-related actions, 675
Pages Pane, 207

Parameters,
  updating, 789
PDF,
  generating via design, 596
Performance, 254
Persistent data, 256
PERSISTENT page source, 1223, 1225, 1228
pie chart features, 914
Pixels as a unit, 984
Placeholder Control,
  and parameter values of control templates, 935
  calling a control template, 935
  overriding control-template actions, 937
PN topics, 664
PostgreSQL,
  connecting natively, 1084
  connecting through ODBC, 1128
postMessage(), 1251
Preview device,
  selecting, 60
  selection of, 204
Print To action, 596
ProcessEmbeddedMessage method,
  of SolutionView control, 1369
Program logo,
  at program start, 1472
Program logo,  
in print output, 1472  

Progress OpenEdge database,  
connecting through JDBC, 1130  
connecting through ODBC, 1132  

Project simulation,  
see Simulation, 73  

Project structure, 120  

Projects, 230  
deploying, 235  
localization of, 250  
location of project files, 233  
namespaces in, 252  
properties, 240  

Purchase action, 668, 1341, 1343  

Purchase action (in example project), 1354  

Push notifications,  
SMT_PUSHNOTIFICATION, 872  
and SMT_PUSHNOTIFICATION tree, 658  
Buttons in, 872  
built-in of, 658  
External PN keys, 662, 872  
how to send, 870, 872  
OnPushNotificationReceived, 872  
overview, 869  
payload of, 658  
PN topics, 664, 872  
receiving mechanism, 872  
Send action, 658  
sending mechanism, 870  
simulation of, 877, 1212  

PXF files,  
for generating PDF, Word, RTF output, 596  

Read Contacts action, 605  
Read Geo Data action, 640  
Recording video, 634  
Register external PN key, 662  
Register PN topics, 664  
Reload action, 689  
(example of use, 1225, 1228  
Replace Node action, 776  
Reset action, 718  
Resolutions, 984  
REST,  
adding as page source, 270  
settings for requests, 270  
REST page source, 267  
Restore Purchases action (in example project), 1359  
Restore Purchases action (iOS only), 669  
Return action, 793  
Reusing page sources, 259  
Rich Text,  
editing in solution, 974  
Rich Text control, 494  
assigning Rich Text style sheet, 967  
editing in solution, 974  
setting up, 966  
Rich Text feature, 966  
Rich Text style sheet,  
assigning to Rich Text control, 967  
creating, 967  
definitions in, 969  
Root node in page source trees,  
context menus of, 301  
RTF,  
generating via design, 596  

Q  
Query Available Products action, 670  
Query Purchases action (Android, Windows), 669  
Query Purchases action (in example project), 1359  
Query/Restore Purchases action, 1343  

S  
Save action, 91, 715  
Save Data setting,  
(example of use, 1223, 1225, 1228  
significance of, 1221  
Save Image to File action, 616  
Save to String, 710  
Saving data to default file, 91  
Scale-independent pixels (sp) as a length unit, 984  
Scroll to Bottom action, 682  
Scrollable tables, 196  

Radio Button control, 482
Send Email To action, 582
Send Push Notification action, 658
Send SMS To action, 590
Serialization, 261, 710
Server,
 levels of interaction with, 231
Server Access setting,
 example of use, 1223, 1225, 1228
 significance of, 1221
Server actions, 748
Server connection errors, 334
Server Services,
 creating a new service, 1373
 deploying a service, 1376
 overview of, 1372
 running a service, 1377
 simulations with trigger data, 1216
Services,
 see Server Services, 1372
Settings,
 3d charts, 925
 area chart features, 914
 bar chart features, 914
 candlestick chart features, 914
 chart background, 912
 chart colors, 918
 chart fonts, 927
 chart grid, 920, 923, 924
 chart legend, 912
 chart title, 912
 chart X-axis, 920
 chart Y-axis, 923
 chart Z-axis, 924
 charts sizes, 925
 for MobileTogether Server, 1474
 for Network Proxy, 1481, 1482
 for simulations, 1475, 1477
 for Trial run on client, 1474
 gauge chart features, 914
 line chart features, 914
 pie chart features, 914
Share action, 587
Show Geolocation on Map action, 644
Show/Hide Wait Cursor action, 605
Signature Field control, 500
Simulating push notifications, 877, 1212
Simulation, 1190
  Debugger modes of, 1233
file locations (designer), 1191
file locations (server), 1196
in MobileTogether Designer, 1191
Messages Pane, 1218
of designs with geolocation components, 1205
of NFC tags for reading data, 1210
of reading calendars from device calendars, 1215
of reading of contacts from device address book, 1214
of service triggers, 1216
on client, 1203
on server, 1196
results and causes, 84
running, 73, 84
with Outlook calendars, 1215
Simulation language for localizations, 1429
Simulation settings,
 access to client apps, 1475
 calendar, 1477
 client GPS information, 1475
 client IP address, 1477
 client/server Push Notifications, 1475
 contacts, 1477
 file folder on client device, 1477
 geolocation, 1477
 NFC, 1477
 Read DB Structure, 1477
 recursive sub pages, 1475
 simulation language, 1475
Sizes and device resolution, 984
SKU ID for in-app purchases, 1335, 1349
SMS, 590
SOAP,
 adding as page source, 279
 settings for requests, 279
SOAP page source, 267
SOAP tutorial, 175
Solution Execution action, 753
Solutions,
 minimize one and start another, 753
 starting another after ending one, 753
Solutions in Windows apps, 1363
SolutionView control,
 running from code, 1368
SolutionView control in XAML file, 1366
SolutionView controls, 1363
Source data,
 see Data sources, 257
sp (scale-independent pixels) as a length unit, 984
| Space control, 510 |
| SQL Server, |
| connecting through ADO, 1064 |
| connecting through ADO.NET, 1070 |
| connecting via JDBC, 1080 |
| SQL statements, 732 |
| SQLite, |
| disable foreign keys, 1087 |
| setting up a connection (Windows), 1087 |
| Start Geolocation Tracking action, 639 |
| Stop Geolocation Tracking action, 639 |
| String to/from XML/JSON, 261, 710 |
| Styles & Properties Pane, 219 |
| StyleVision Server, |
| for generating PDF, Word, RTF output, 596 |
| Sub pages, 207 |
| SubPages tutorial, 148 |
| Suggested image file extension, 616 |
| Switch control, 512 |
| Sybase, |
| connecting through JDBC, 1134 |
| Synchronize offline data with online databases, 1221, 1223, 1225, 1228 |
| System DSN, |
| setting up, 1077 |
| System requirements, 48 |

**T**

| Tab order, |
| defining controls in sequence, 1442 |
| Tab splits, 207 |
| Tabbed pages, 207 |
| Table, |
| adding, 64 |
| Table border settings, 1452 |
| Table column widths, 1472 |
| Table control, 523 |
| Tables, 809 |
| and databases, 129, 135, 141 |
| context menu, 831 |
| dynamic columns in, 820 |
| dynamic rows and columns in, 820 |
| dynamic rows in, 816 |
| properties, 824 |
| repeating, 812 |
| scrollable, 196 |
| static, 810 |
| Telephony, 591 |
| Template Event Callback action, 937 |
| Teradata, |
| connect through JDBC, 1136 |
| connect through ODBC, 1138 |
| Terminology, 50 |
| Test cases, |
| deploying to server, 1298 |
| playing back, 1291 |
| recording, 1289 |
| Test cases and runs, |
| managing, 1294 |
| Test runs, |
| comparing, 1302 |
| Testing the design, |
| see Simulation, 1203 |
| Text content of mixed-content elements, 1038 |
| Text size, |
| auto-fit groups listing, 1444 |
| Text string, |
| exporting from node to text file, 702 |
| importing from text file to node, 702 |
| Throw action, 789 |
| Time, |
| user selection of, 612 |
| Time control, 549 |
| Top pages, 207 |
| Tracepoints in XPath Debugger, 996 |
| Transforming images, 850 |
| Tree data, 289 |
| default file for, 296 |
| importing data from file, 296 |
| manually entering, 296 |
| Tree nodes in page source trees, |
| context menus of, 301 |
| Tree structure, 289 |
| about, 294 |
| manually creating, 294 |
| Trial run on client, 1203 |
| Try/Catch Exceptions action, 791 |
| Try/Catch Server Connection Error action, 793 |
| Tutorials, 56 |
| Database and Charts tutorial, 119 |
| file locations, 58, 79 |
| QuickStart (Part 1), 58 |
| QuickStart (Part 2), 79 |
Tutorials, 56
  SOAP, 175
  SubPages, 148
  Visibility, 148

U

Update data actions, 760
  Update Display action, 684
  Update Node action, 762
  Update Node(s),
    example of use, 1225
  Update Node(s) action,
    example of use, 1223, 1228
  Update Variable action, 789
  Updating variables and parameters, 789
  User Cancel Behavior action, 751
  User DSN,
    setting up, 1077
  User interface,
    description of mechanisms, 201
    overview, 201
  User login, 1255
  User variables, 1050
  User-defined XPath/XQuery functions, 1035

W

WADL,
  in REST requests, 270
Wait Cursor, 605
Watch expressions in XPath Debugger, 996
Watches in action debugging, 1234
Window title, 1472
Windows apps,
  see UWP apps, 1363
Word,
  generating via design, 596
Workflow, 120
  defined by sequence of top pages, 207
Workflow simulation,
  see Simulation, 73
Working Directory setting on server,
  and server simulation, 1196
WSDL,
  in SOAP requests, 279

X

XAML file for UWP app, 1366
XML data serialization, 261, 710
XML page source, 261
XML to/from string, 261, 710
XML trees, 289
XPath context node, 61
XPath Debugger, 1232
  description of, 996
    list of all ways to access, 1242
    opening directly from Simulator, 1242
  XPath expressions, 987
XPath expressions, 987
  debugging, 996
  uses in MobileTogether designs, 53
XPath/XQuery,
  FAQ, 1038
XPath/XQuery Expression Builder, 988, 989
XPath/XQuery Expression Dialog, 988
XPath/XQuery Expression Evaluator, 988, 992
XPath/XQuery functions,
  user-defined, 1035
XQuery expressions, 987
XQuery page source, 282